English Abstract for Reference AA : CN1360414

AA

Abstract not available for CN1360414

Abstract of corresponding document: **EP1220563**

An optical switch network (4) includes optical routers (10), which route information in optical fibers (12). Each fiber carries a plurality of data channels (20), collectively a data channel group (14), and a control channel (16). Data is carried on the data channels in data bursts and control information is carried on the control channel (18) in burst header packets. A burst header packet includes routing information for an associated data burst (28) and precedes its associated data burst. Parallel scheduling at multiple delays may be used for faster scheduling. In one embodiment, unscheduled times and gaps may be processed in a unified memory for more efficient operation.

FIG. 1a

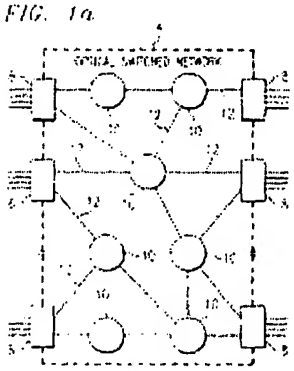
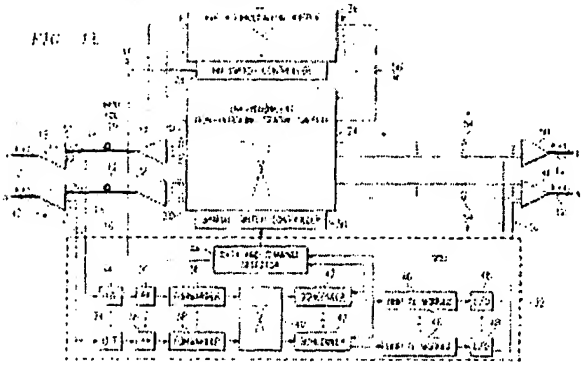


FIG. 1b



IN032451

[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
H04J 14/02
H04B 10/12 H04Q 3/52

[12]发明专利申请公开说明书

[21]申请号 01133873.3

[43]公开日 2002年7月24日

[11]公开号 CN 1360414A

[22]申请日 2001.12.21 [21]申请号 01133873.3

[30]优先权

[32]2000.12.22 [33]US [31]60/257,487

[71]申请人 阿尔卡塔尔公司

地址 法国巴黎

[72]发明人 熊一俊 郑斯清

[74]专利代理机构 中国国际贸易促进委员会专利商标事务所

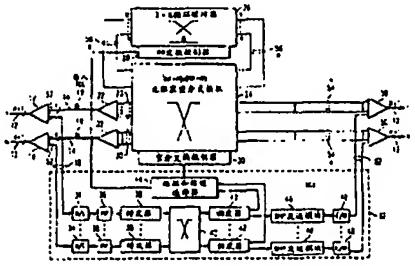
代理人 李德山

权利要求书2页 说明书38页 附图页数13页

[54]发明名称 光路由器中的信道调度

[57]摘要

本发明涉及光路由器中的数据信道调度器的统一关联存储器。光交换网络(4)包括光路由器(10),它以光纤(12)发送信息。每根光纤都载送多个数据信道(20)和一个控制信道(16),其中多个数据信道可集中成数据信道群(14)。数据在数据信道上以数据子帧方式被载送,而控制信息在控制信道(18)上以脉冲串标题分组形式被载送。脉冲串标题分组包括关联数据子帧(28)的路由信息,并且先于其关联数据子帧。多延迟并行调度可以实现快速调度。在一种实施方式中,为了更有效地操作,可以在统一存储器中处理未调度时间和间隙。



ISSN1008-4274

知识产权出版社出版

权 利 要 求 书

1. 一种用于调度光成组交换路由器中的数据子帧的电路，包括：

一个光交换机，用于将来自入口光传输媒体的光信息路由发送到多个出口光传输媒体之一；

一个与所述光交换机连接的延迟缓冲器，用于提供n个不同的延迟，以便延迟所述入口传输媒体与所述出口光传输媒体之间的信息；

与各个的出口媒体有关联的调度电路，它包括n+1个关联处理器，每个关联处理器包括电路用于：

为关联出口光传输媒体存储关于n个延迟中相应的一个和零延迟的调度信息；

关于各个延迟识别可用的时间段，其间一个数据子帧可被调度。

2. 权利要求 1 的电路，其中，所述入口光传输媒体和出口光传输媒体包括光纤。

3. 权利要求 1 的电路，其中，所述关联存储器识别未调度时间段。

4. 权利要求 1 的电路，其中，关联处理器识别调度的数据子帧之间的间隙。

5. 权利要求 4 的电路，还包括第二组 n+1 个关联处理器，其中第二组关联处理器识别未调度时间段。

6. 权利要求 1 的电路，其中所述延迟缓冲器包括独立的延迟线，其中每个延迟线连接所述光交换机的一个预定输入和一个预定输出。

7. 权利要求 1 的电路，其中所述延迟缓冲器包括一个延迟线矩阵，其中每个期望的延迟线可被连接在所述光交换机的一个选定输入和一个选定输出之间。

8. 一种调度光成组交换路由器中的数据子帧的方法，该路由器通过一个光交换机从一个入口光传输媒体向多个光传输媒体之一或者直接通过该光交换机或者经由一个延迟缓冲器的 n 个不同延迟之一路由发送光信息，该方法包括如下步骤：

在n+1个关联存储器中，对于每一相关的出口光传输媒体，存储关于n个延迟中相应一个和一个零延迟的调度信息；和

当前识别在其中可调度数据子帧的每个所述关联处理器中可用的时间段，以便能够同时确定与多个延迟相关的可用时间段。

9. 权利要求 8 的方法，其中，所述入口光传输媒体和出口光传输媒体包括光纤.

10. 权利要求 8 的方法，其中，所述当前识别步骤包括当前识别在每个关联处理器中的未调度时间段的步骤.

11. 权利要求 8 的方法，其中，所述当前识别步骤包括当前识别在每个所述关联处理器中的数据子帧之间的间隙的步骤.

12. 权利要求 11 的方法，其中，所述当前识别步骤包括当前识别在每个所述关联处理器中的未调度时间段的步骤.

13. 权利要求 8 的方法，其中，所述延迟缓冲器包括单独的延迟线，每个延迟线连接所述光交换机的一个预定输入和一个预定输出.

14. 权利要求8的方法，其中，所述延迟缓冲器包括一个延迟线矩阵，其中每个期望的延迟线可被连接在所述光交换机的一个选定输入和一个选定输出之间。

说　　明　　书

光路由器中的信道调度

相关申请交叉参考

本申请要求共同未决美国临时申请系列号 60/257,884 的申请日，该临时申请由 Zheng 等人于 2000 年 12 月 22 日申请，名称为“光路由器中的数据信道调度器的统一关联存储器”

技术领域

本发明涉及电信领域，尤其涉及一种光交换的方法和设备。

背景技术

近来，网络中尤其是因特网中的数据业务量急剧增加，并且随着用户的增多和有宽带宽要求的新业务的采用，数据业务量将继续增加。因特网业务量的增加需要具有高容量路由器的网络，以便能够发送可变长度的数据分组。一种选择是采用光网络。

新出现的密集波分复用（DWDM）技术通过提高光纤的容量改善了带宽问题。然而，这种提高的容量造成了与现有电子交换技术的严重失配，因为与 DWDM 的每秒若干兆兆比特容量相比，现有电子交换技术只能交换速率最多到每秒几千兆比特的数据。虽然新兴 ATM 交换机和 IP 路由器可利用光纤中的单个信道通常以每秒几百千兆来交换数据，然而，这种方法意味着必须要用几十或几百个交换接口来端接具有大量信道的单根 DWDM 光纤。这样，当并行信道只用作独立链路的集合而不是用作共享资源时，可能导致很大的统计复用效率的损失。

目前，提出了各种主张在交换系统中采用光学技术来取代电子技术的方法；然而，光学部件技术的局限性大大地限制了光交换的灵巧管理/控制应用。一种方法，称为光成组交换联网，试图最佳利用光和

电交换技术。电子技术通过将单独的用户数据脉冲串分配给 DWDM 光纤的信道来动态控制系统资源，而光学技术只用于在光学域中交换用户数据信道。

以前的用于直接处理端对端用户数据信道的光网络已不太令人满意.

因此，需要一种用于提供光成组交换网络的方法和设备。

发明内容

本发明提供一种用于调度光成组交换路由器中的数据子帧的电路。一个光交换机将来自入口光传输媒体的光信息路由发送到多个出口光传输媒体之一。一个与所述光交换机连接的延迟缓冲器提供 n 个不同的延迟，以便延迟入口传输媒体与出口光传输媒体之间的信息。调度电路与各个出口媒体关联；每个调度电路包括 n+1 个关联处理器。每个关联处理器包括电路用于：（1）为关联的出口光传输媒体存储关于 n 个延迟中相应的一个和零延迟的调度信息；和（2）关于各个延迟识别可用的时间段，其间一个数据子帧可被调度。

本发明提供快速而有效的方法用于调度在光成组交换路由器中的脉冲串.

附图说明

为了更好地理解本发明及其优点，下面将参照结合附图所作的说明，其中:

图 1a 是光网络的框图;

图 1b 是核心光路由器的框图;

图 2 示出了调度过程的数据流;

图 3 示出了调度器的框图;

图 4a 和 4b 示出了脉冲串标题分组相对于数据子帧的到达时序图;

图 5 示出了 DCS 模块的框图;

图 6 示出了关联存储器 PM 的框图;

图 7 示出了关联存储器 PG 的框图;

图 8 示出了 LAUC-VF 调度方法的流程图;

图 9 示出了 CCS 模块的框图;

图 10 示出了关联存储器 P_T 的框图;

图 11 示出了调度控制信道的受限定的早期方法的流程图;

图 12 示出了路径和信道选择器的框图;

图 13 示出了一例通过循环缓冲器的阻塞输出信道;

图 14 示出了 BHP 传输模块的存储器的存储结构;

图 15 示出了采用无源 FDL 环路的光路由器体系结构的框图;

图 16 示出了一例具有多个 P_M 和 P_G 对的路径和信道调度器的框图;

图 17a 和 17b 示出了输出数据信道的时序图;

图 18 示出了 CLK_f 和 CLK_s 的时钟信号;

图 19a 和 19b 示出了用于路由器的时隙操作的可选硬件改进方式;

图 20 示出了关联处理器 P_M 的框图;

图 21 示出了关联处理器 P_G 的框图;

图 22 示出了关联处理器 P_{MG} 的框图;

图 23 示出了关联处理器 P^*_{MG} 的框图;

图 24 示出了一种利用多个关联处理器来快速调度的实施方式的框图;

图 25 示出了供多个信道群使用的处理器 $P_{M\text{-}ext}$ 的框图; 和

图 26 示出了供多个信道群使用的处理器 $P_{G\text{-}ext}$ 的框图.

具体实施方式

结合附图中的图 1-26, 能很好地理解本发明, 各附图中, 用相同的标号表示相同的部分.

图 1a 示出了光成组交换网络 4 的一般框图. 光成组交换 (OBS) 网络 4 包括多个电子入口端路由器 6 和多个出口端路由器 8. 这些入口端路由器 6 和出口端路由器 8 与多个核心光路由器 10 连接. 入口端

路由器 6、出口端路由器 8 和核心路由器 10 之间利用光链路 12 进行连接. 每根光纤能够载送光数据的多个信道。

工作时，光数据的数据子帧（或简称为“脉冲串”）是准备通过网络 4 发送的基本数据块. 入口端路由器 6 和出口端路由器 8 负责脉冲串的组合和拆分功能，并作为光成组交换网络 4 与常规电子路由器之间的老式接口.

在光成组交换网络 4 中，所要发送的基本数据块是脉冲串，它是一些具有某些共同属性的分组的集合. 脉冲串由脉冲串有效载荷（称之为“数据子帧”）和脉冲串标题（称之为“脉冲串标题分组”或 BHP）组成. 光成组交换网络的固有特征在于，在每个网络节点中，数据子帧及其 BHP 以不同的信道传送，并且分别在光学域和电子域中进行交换. BHP 比其关联数据子帧提前一个时间偏移量$\tau(\geq 0)$被发送. 其初始值τ_0由（电子）入口端路由器 8 设置.

本发明中，“信道”被定义为两个邻近路由器之间的某一单向传输能力（以每秒比特为单位）. 一个信道可以由一个波长或几分之一波长组成（例如，当采用时分复用时）. 载送数据子帧的信道被称为“数据信道”，而载送 BHP 和其他控制分组的信道被称为“控制信道”。“信道群”是一组具有共同类型并且节点邻近的信道. 链路被定义为两个路由器之间的总传输能力，通常由每个方向上的“数据信道群”（DCG）和“控制信道群”（CCG）组成.

图 1b 示出了核心光路由器 10 的框图. 对于每根光纤 12，分用器 18 将入口 DCG 14 与 CCG 16 分离开. 每个 DCG 14 被光纤延迟线（FDL）19 所延迟. 分用器 22 将延迟后的 DCG 分为信道 20. 每个信道 20 被输入到无阻塞空分交换机 24 上各自的输入节点. 空分交换机 24 的另外一些输入和输出节点与循环缓冲器（RB）26 连接. 循环缓冲器 26 由循环交换控制器 28 控制. 空分交换机 24 由空分交换控制器 30 控制.

CCG 16 与一个交换控制单元（SCU）32 连接. 针对各 CCG 16，SCU 包括一个光/电收发信机 34. 光/电收发信机 34 接收光 CCG 控制

信息并将该光信息转换成电信号. 电 CCG 信息被分组处理器 36 所接收，该处理器再将信息传送到转发器 38. 各 CCG 的转发器与交换机 40 连接. 交换机 40 的输出节点与各自的调度器 42 连接. 调度器 42 与路径和信道选择器 44 连接，还与各自的 BHP 发送模块 46 连接. BHP 发送模块 46 与电/光收发信机 48 连接. 电/光收发信机产生输出 CCG 52，以便通过复用器 50 与各自的输出 DCG 54 信息相关联. 路径和信道选择器 44 还与 RB 交换控制器 28 和空分交换控制器 30 连接.

图 1b 中所示的实施方式具有 N 个输入 DCG-CCG 对和 N 个输出 DCG-CCG 对 52，其中每个 DCG 都有 K 个信道而每个 CCG 只有一个信道（k=1）. DCG-CCG 对 52 以一根光纤来载送. 一般而言，光路由器可能是不对称的，CCG 16 的信道数 k 可能大于 1，而 DCG-CCG 对 52 可能以一根以上的光纤 12 来载送. 在所示的实施方式中，有一个缓冲信道群（BCG）56，该信道群具有 R 个缓冲信道. 一般而言，可以有一个以上的 BCG 56. 光交换矩阵（OSM）包含一个(NK+R)×(NK+R)空分交换机和一个具有 WDM（波分复用）FDL 缓冲器的 R×R 交换机，其中缓冲器用作循环缓冲器（RB）26，以解决出口数据信道上的数据子帧争用问题. 空分交换机是一个绝对无阻塞交换机，这意味着，入口数据信道上到达的数据子帧可以被交换到任一空闲的出口数据信道. 输入 FDL 19 所引入的延迟 Δ 应当足够长，以便 SCU 32 在其关联数据子帧进入到空分交换机之前有足够的时间来处理 BHP.

R×R RB 交换机是一种广播和选择类型的交换机，这种类型如“P.Gambini 等，‘Transparent Optical Packet Switching Network Architecture and Demonstrators in the KEOPS Project’，*IEEE J.Selected Areas in Communications*, vol.16, no.7, pp.1245-1259, Sept.1998”中所述. 假定，R×R RB 交换机有 B 根 FDL，其中第 i 根 FDL 引入 Q_i 延时，$1 \leqslant i \leqslant B$. 在不失一般性情况下，还假定，$Q_1<Q_2<\ldots<Q_B$ 和 $Q_0=0$（意味着没有使用 FDL 缓冲器）. 注意，所有 N 个输入 DCG 共同使用 FDL 缓冲器，而每根 FDL 包括 R 个信道. 以任意入口信道进入 RB 交换机的数据子帧可被所提供的 B 个延时之

一所延时. 图 1b 中的循环缓冲器通过去掉 RB 交换机的功能可被简化为无源 FDL 环路，如图 15 中所示，其中不同的缓冲信道可以具有不同的延迟.

SCU 部分基于电子路由器. 在图 1b 中，SCU 具有 N 个输入控制信道和 N 个输出控制信道. SCU 主要包括：分组处理器（PP）36，转发器 38，交换结构 40，调度器 42，BHP 传输模块 46，路径和信道选择器 44，空分交换控制器 30，和 RB 交换控制器 28. 分组处理器 36，转发器 38 和交换结构 40 可以存在于电子路由器中. 其他部件（尤其是调度器）对光路由器而言是新出现的. 设计 SCU 时尽可能多地采用分布式控制，但对被集中的共用 FDL 缓冲器的存取的控制除外.

分组处理器执行第一层和第二层拆封功能并为每个到达的 BHP 附上一个时间标记，这样可以记录关联数据子帧到达 OSM 的时间. 该时间标记是BHP到达时间、BHP所载送的脉冲串偏移时间τ和输入FDL 19 所引入的延迟Δ的总和. 转发器主要执行转发一览表，以判断哪个出口 CCG 52 转发 BHP. 关联数据子帧将被交换到相应的 DCG 54. 可以用无连接或面向连接方式来完成转发.

每个 DCG-CCG 对 52 都有一个调度器. 调度器 42 根据 BHP 所载送的信息调度出口 DCG 54 的数据信道上的数据子帧的交换. 如果发现有空闲的数据信道，那么调度器 42 将调度出口控制信道上的 BHP 的传输，从而通过保持偏移时间τ（≥ 0）尽可能接近τ_0，尝试使 BHP 与其关联数据子帧再同步. 在顺利地调度了数据子帧和 BHP 之后，如果调度器 42 不必通过循环缓冲器 26 提供延迟，那么它将配置信息发送给空分交换控制器 30，否则，调度器还要将配置信息发送给 RB 交换控制器 28.

图 2 中示出了调度判决过程的数据流. 在判决块 60 中，调度器 42 判断是否有足够的时间来调度入口数据子帧. 如果是的话，那么，调度器判断是否可以调度数据子帧，即在指定输出 DCG 54 中是否有空闲空间用于数据子帧. 为了调度数据子帧，必须有可用的空间提供给指定输出 DCG 中的数据子帧. 这一空间可以在以数据子帧到空分交

换机 24 的到达时间为起点延长到循环缓冲器 26 所能提供的最大延迟的时间窗内起动。如果可以调度数据子帧，那么，在判决块 64 中，调度器 42 必须判断在输出 CCG 52 中是否有可用的空间用于 BHP.

如果判决块 60、62 或 64 中的任一判决为否定的话，那么在块 65 中放弃数据子帧和 BHP. 如果判决块 60、62 和 64 中的所有判决均为肯定的话，那么调度器向路径和信道选择器 44 发送调度信息. 从调度器到路径和信道选择器的配置信息包括：入口 DCG 标识符，入口数据信道标识符，出口 DCG 标识符，出口数据信道标识符，数据子帧到空分交换机的到达时间，数据子帧时长，FDL 标识符 i（要求 Q_i 延时，$0 \leqslant i \leqslant B$）.

如果 FDL 标识符为 0，也就是说无需 FDL 缓冲器，那么，路径和信道选择器 44 简单地将配置信息转发给空分交换控制器 30. 否则，在判决块 68 中，路径和信道选择器 44 寻找发向 RB 交换机 26 的空闲入口缓冲器信道. 如果有的话，那么在判决块 70 中，路径和信道选择器 44 寻找发自 RB 交换机 26 的空闲出口缓冲器信道，以便在 RB 交换机 26 中的指定延迟之后载送重新进入空分交换机的数据子帧。假定，一旦数据子帧进入 RB 交换机，它就可以被延迟集合$\{Q_1,Q_2,\ldots,Q_B\}$中的任一离散时间. 如果情况不是这样，那么路径和信道选择器 44 必须得考虑 RB 交换机体系结构. 如果进出 RB 交换机 26 的空闲信道都能找到，那么，路径和信道选择器 44 将配置信息发送给空分交换控制器 30 和 RB 交换控制器 28，并向 42 调度器发回一个 ACK（确认）. 否则，它向调度器 42 发回一个 NACK（否认），并在块 65 中放弃 BHP 和数据子帧.

从路径和信道选择器 44 到空分交换控制器 30 的配置信息包括：入口 DCG 标识符，入口数据信道标识符，出口 DCG 标识符，出口数据信道标识符，数据子帧到空分交换机的到达时间，数据子帧时长，FDL 标识符 i（要求 Q_i 延时，$0 \leqslant i \leqslant B$）. 如果 $i>0$，那么，该信息还包括：入口 BCG 标识符（到 RB 交换机），入口缓冲信道标识符（到 RB 交

换机），出口 BCG 标识符（来自 RB 交换机），和出口缓冲信道标识符（来自 RB 交换机）.

从路径和信道选择器到RB交换控制器的配置信息包括:入口 BCG 标识符（到 RB 交换机），入口缓冲信道标识符（到 RB 交换机），出口 BCG 标识符（来自 RB 交换机），出口缓冲信道标识符（来自 RB 交换机），数据子帧到 RB 交换机的到达时间，数据子帧时长，FDL 标识符 i（要求 Q_i 延时，$1 \leqslant i \leqslant B$）.

空分交换控制器 30 和 RB 交换控制器 28 可执行从所接收到的配置信息到建立内部路径时所涉及的物理组成的映射，并且可使交换机及时地使数据子帧快速通过光路由器 10. 当 FDL 标识符大于 0 时，空分交换控制器可以在空分交换机中建立两个内部路径，一个是当数据子帧到达空分交换机时从入口数据信道到入口循环缓冲信道的路径，另一个是当数据子帧重新进入空分交换机时从出口缓冲信道到出口数据信道的路径. 当接收到来自路径和信道选择器 44 的 ACK 时，调度器 42 可更新所选择的数据和控制信道的状态信息，并准备处理新的 BHP.

最后，BHP 传输模块在调度器所指定的时间传输 BHP.

以上是关于在光路由器中如何调度数据子帧的一般描述. 必要的话，可以根据下述设计原理容易地推广通过 $R \times R$ 循环缓冲交换机循环数据帧一次以上.

图 3 示出了调度器 42 的框图. 调度器 42 包括: 调度队列 80，BHP 处理器 82，数据信道调度(DCS)模块 84，和控制信道调度(CCS)模块 86. 每个调度器只需保持对其关联出口 DCG 54 和出口 CCG 52 的忙/闲时段的跟踪.

来自电子交换机的 BHP 首先保存在调度队列 80 中. 对于基本操作，所需要的是一个调度队列 80，然而对于不同的业务级别，可以保留一些虚拟调度队列 80. 可以根据 BHP 的到达次序或根据其关联数据子帧的实际到达次序来提供各个队列 80. BHP 处理器 82 协调数据和控制信道的调度过程，并将配置情况发送给路径和信道选择器 44.

它可以顺序地或并行地启动 DCS 模块 84 和 CCS 模块 82，这取决于 DCS 和 CCS 模块 84 和 82 的实现情况.

在串行调度情况下，BHP 处理器 82 首先启动 DCS 模块 84，以便在所要求的输出 DCS 54 中调度数据信道上的数据子帧（DB）. 在判定了何时发出数据子帧之后，BHP 处理器便启动 CCS 模块 86，以便调度关联控制信道上的 BHP.

在并行调度情况下，BHP 处理器 82 同时启动 DCS 模块 84 和 CCS 模块 86. 由于 CCS 模块 86 不知道何时发出数据子帧，因此，它针对数据子帧的所有可能的出发时间或其子集来调度 BHP. 总共有 B+1 种可能的出发时间. 根据 DCS 模块 84 所报告的实际数据子帧出发时间，BHP 处理器 86 将选出合适的时间来发出 BHP.

在 OBS 网络中的边缘与核心之间以及核心节点之间的数据和控制信道中，可采用时隙传输. 时隙是一个固定长度的时间段. 设 T_s 是数据信道中一个时隙的时长（例如，以 μs 为单位），而 T_f 是控制信道中一个时隙的时长. 如果数据信道速率是每秒 r_d 千兆比特，那么一个时隙可发送 T_s*r_d Kbit 信息. 同样，如果控制信道速率是每秒 r_c 千兆比特，那么一个时隙可发送 T_f*r_c Kbit 信息. 考虑两种设想：(1)$r_c=r_d$ 和(2)$r_c \neq r_d$. 在后一种情况下，一个典型的例子是 $r_c=r_d/4$（例如，OC-48 用于控制信道，而 OC-192 用于数据信道）.

在不失一般性的情况下，假定，T_f 等于 T_s 的若干倍. 图 4a 和 4b（也可参见图 18）中描述了两个例子，这里说明当初始偏移时间 $\tau_0=8T_s$ 时，在 $T_f=T_s$ 以及 $T_f=4T_s$ 的情况下时隙传输系统中的时间标记和脉冲串偏移时间. 为了便于描述，我们用时*频*来表示控制信道中的时*隙*. 在不失一般性的情况下，还假定：(1)数据子帧长度可变，为时隙的倍数，它们可以只到达时隙界线，和(2)BHP 长度也可变，为字节的倍数. 固定长度的数据子帧和 BHP 仅仅是特例. 在时隙传输中，针对不同的用途，在各时隙中都有一些额外开销，比如同步和检错. 假定，控制信道上的帧有效载荷为 P_f 个字节，它小于（T_f*r_c）*1000/8 字节，因此，在时帧中可以发送总信息量.

OSM 具有周期性. 对于数据信道上的时隙传输, 配置周期的一个典型的例子是 1 个时隙, 尽管配置周期也可能是时隙的倍数. 这里, 假定每个时隙都对 OSM 进行配置. FDL Q_i 的长度也要求是时隙的倍数, $1 \leqslant i \leqslant B$. 由于时隙传输和交换, 建议在 SCU 中使用时隙作为*基本时间单位*, 以便于数据信道调度、控制信道调度和缓冲信道调度, 以及 BHP 与其关联数据子帧之间的同步. 这样便于设计各种调度器.

关于图 4a、4b 和 5, 将使用下列整数变量:

t_{BHP}: BHP 进入 SCU 的时帧的起点;

t_{DB}: 数据子帧 (DB) 到达光交换矩阵 (OSM) 的到达时间;

l_{DB}: 以时隙为单位的 DB 的时长/长度;

Δ: 输入 FDL 所引入的延迟 (以时隙为单位);

τ: 脉冲串偏移时间 (以时隙为单位).

每个到达 SCU 的 BHP 在收发信机接口处经 O/E 转换后都被标上时间标记, 以便记录 BHP 进入 SCU 的时帧的起点. 对于 SCU 在同一时帧接收到的 BHP, 它们将具有相同的时间标记 t_{BHP}. 为了便于调度, 最重要的变量是 t_{DB} (即 DB 到达 OSM 的到达时间). 假定在 SCU 中使用 b-bit 时隙计数器来跟踪时间, 那么可以按下式来计算 t_{DB}:

$$t_{DB}=(t_{BHP} \cdot T_f+\Delta+\tau) \bmod 2^b \qquad (1)$$

在 SCU 32 中, 可以由 BHP 来载送时间标记 t_{DB}. 注意, 也可以从 BHP 到达的时帧的起点开始计算脉冲串偏移时间 τ, 如图 4a-b 中所示, 其中, 在图 4a 中, $t_{BHP}=9$ 和 $\tau=6$ 个时隙, 而在图 4b 中, $t_{BHP}=2$ 和 $\tau=7$ 个时隙. 假定 $\Delta=100$ 个时隙, 则有 $t_{DB}=115$, 即 DB 将在时隙界线 115 到达. 在图 4a-b 中, $1 \leqslant \tau \leqslant \tau_0=8$. 在不失一般性的情况下, 假定, 图 1b 中的空分交换机的交换等待时间可以忽略. 因此, 如果没有使用 FDL 缓冲器的话, 那么, 数据子帧到达空分交换机 24 的到达时间 t_{DB} 也就是其出发时间. 注意, 即便交换等待时间不可忽略, 在信道调度时, t_{DB} 仍可以作为数据子帧出发时间, 这是因为, 在路由器输出端口可以补偿交换等待时间, 从而在此使数据和控制信道再同步.

图 5 示出了 DCS 模块 84 的框图. 在这一实施方式中，关联处理器阵列 P_M 90 和 P_G 92 并行搜索未调度信道时间以及调度信道时间之间的间隙，并更新状态信息. 间隙和未调度时间以相对时间来表示. P_M 90 和 P_G 92 与控制处理器 CP_1 94 连接. 在一种实施方式中，采用 LAUC-VF（具有空填充的最近有效未使用信道）调度原理来确定所需的调度，这种调度原理如 Zheng 等人于 2000 年 10 月 12 日申请的美国系列号 09/689,584 中所述，该申请的名称为“具有 FDL 缓冲器的光路由器的信道调度算法的硬件实现”，在此作为参考.

DCS 模块 84 使用两个 b-bit 时隙计数器 C 和 C_1. 计数器 C 跟踪与 CCS 模块 86 共同使用的时隙. 计数器 C_1 记录自从接收到最近的 BHP 所逝去的时隙. 这两个计数器以时隙时钟的脉冲而递增. 然而，当 DCS 模块 84 接收到新的 BHP 时，计数器 C_1 复位到 0. 一旦计数器 C_1 达到 2^b-1，它就停止计数，这表示自从接收到最近的 BHP 后已逝去了至少 2^b-1 个时隙. b 的值应当满足 $2^b \geqslant W_s$，其中 W_s 是数据信道调度窗. $W_s=\tau_0+\Delta+Q_B+L_{max}-\delta$，其中 L_{max} 是 DB 的最大长度，而 δ 是 BHP 从 O/E 转换器到调度器 42 的最小延迟. 假定，τ_0=8，Δ=120，Q_B=32，L_{max}=64，而 δ=40，那么，W_s=184 个时隙. 在这种情况下，b=8 比特.

图 5 中的关联处理器 P_M 用来保存 DCG 中的各数据信道的未调度时间. 设 t_i 为信道 H_i 的未调度时间，它被保存在 P_M 的第 i 个条目中，$0 \leqslant i \leqslant K-1$. 于是，从时隙 t_i 向前，信道 H_i 是空闲的，即没有被调度. t_i 是一个相对于调度器接收到最近 BHP 的时隙的相对时间. P_M 有一个 2K 字的关联存储器，用以分别保存未调度时间 t_i 和信道标识符. 未调度时间按下降次序被保存. 例如，在图 6 中，有 K=8 和 $t_0 \geqslant t_1 \geqslant t_2 \geqslant t_3 \geqslant t_4 \geqslant t_5 \geqslant t_6 \geqslant t_7$.

类似地，图 5 中的关联处理器 P_G 用来保存 DCG 中的各数据信道的间隙. 我们用 l_j 和 r_j 来表示间隙 j 的起始时间和结束时间，$0 \leqslant j \leqslant G-1$，这些时间也是相对时间. 这一时隙被保存在 P_G 的第 j 个条目中，其相应的数据信道是 H_j. P_G 有一个 G 个字的关联存储器，用以分别保存间隙起始时间、间隙结束时间和信道标识符. 间隙也可以根据其

起始时间 l_j 按下降次序被保存. 例如，图 7 示出了这种关联存储器 P_G，其中 $l_0 \geq l_1 \geq l_2 \geq \ldots \geq l_{G-2} \geq l_{G-1}$. G 是可以被保存的间隙的总数. 如果间隙多于 G 个，那么具有较大起始时间的最新间隙将推出具有最小起始时间的间隙，它驻留在关联存储器的底部. 注意，如果 $l_j=0$，那么，在 DCG 中总共有 j 个间隙，因为 $l_{j+1}=l_{j+2}=\ldots=l_{G-1}=0$.

一旦接收到 BHP 处理器发出的调度出发时间为 t_{DB} 而时长为 l_{DB} 的 DB 的请求，控制处理器（CP_1）94 首先就记录下接收到该请求的时隙 t_{sch}，读取计数器 C_1（$t_e \leftarrow C_1$），并将 C_1 复位到 0. 然后，CP_1 将 t_{sch} 作为新的参考时间，计算出相对于 t_{sch} 的 DB 出发时间（无 FDL 缓冲器）为:

$$t'_{DB} = (t_{DB} - t_{sch} + 2^b) \bmod 2^b, \qquad (2)$$

同时，CP_1 根据下式更新 P_M:

$$t_i = \max(0, t_i - t_e), \qquad 0 \leq i \leq K-1 \qquad (3)$$

然后，利用以下公式更新 P_G:

$$l_j = \max(0, l_j - t_e), \qquad 0 \leq j \leq G-1 \qquad (4)$$

和

$$r_j = \max(0, r_j - t_e), \qquad 0 \leq j \leq G-1. \qquad (5)$$

存储器更新之后，CP_1 94 搜索合格的出口数据信道，以便根据以上所引用的 LAUC-VF 方法载送数据子帧. 流程图如图 8 中所示. 在块 100 中，指数 i 设为“0”. 在块 102 中，P_G 找出一个间隙以便在 $t'_{DB}+Q_i$ 发送数据子帧. 在块 106 中，P_M 在 P_M 中找出未调度信道以便在 $t'_{DB}+Q_i$ 时间发送数据子帧. 注意，在 P_G 中找出间隙以便在时间 $t'_{DB}+Q_i$ 发送 DB 的操作和在 P_M 中找出未调度时间以便在时间 $t'_{DB}+Q_i$ 发送 DB 的操作最好并行执行. 在 P_G 中找出间隙以便在时间 $t'_{DB}+Q_i$ 发送数据子帧的操作（块 102）包括将 P_G 中的每个条目与（$t'_{DB}+Q_i$，$t'_{DB}+Q_i+l_{DB}$）

并行比较. 如果 $t'_{DB}+Q_i \geq l_j$ 而 $t'_{DB}+Q_i+l_{DB} \leq r_j$，那么，条目 j 的响应比特返回 1，否则，它返回 0，$0 \leq j \leq G-1$. 如果 P_G 中至少一个条目返回 1，那么选择最小指数的间隙.

在 P_M 中找出未调度时间以便在时间 $t'_{DB}+Q_i$ 发送 DB 的操作（块 106）包括将 P_M 中的每个条目与 $t'_{DB}+Q_i$ 并行比较. 如果 $t'_{DB}+Q_i \geq t_j$，那么，条目 j 的响应比特返回 1，否则，它返回 0，$0 \leq j \leq K-1$. 如果 P_M 中至少一个条目返回 1，那么选择最小指数的条目.

如果在判决块 104 或 108 中调度成功，那么，在块 105 或 109 中，CP_1 分别将所选择的出口数据信道和 FDL 标识符通知给 BHP 处理器 82. 在接收到来自 BHP 处理器 82 的 ACK 之后，CP_1 94 将更新 P_G 90 或 P_M 94 或者两者都更新. 如果调度不成功，那么在块 110 中，将 i 递增，并且 P_M 和 P_G 试验某一时间，以便以不同的延迟调度数据子帧. 一旦 Q_i 达到最大延迟（判决块 112），处理器 P_M 和 P_G 在块 114 中报告无法调度数据子帧.

为了提高调度处理过程的速度，可以并行进行搜索. 例如，如果 B=2，并使用了三个相同的 P_M'和 P_G'，如图 5 中所示，那么，一个并行搜索将判断是否可以在时间 t'_{DB}、$t'_{DB}+Q_1$ 和 $t'_{DB}+Q_2$ 发出数据子帧. 如果能以不同的时间发出数据子帧，那么选择最小的时间. 在另一例中，如果 B=5，并使用了三个相同的 P_M'和 P_G'，那么，至多两个并行搜索将判断是否可以调度 DB.

下面列举了 LAUC-VF 方法的一些简化形式，它们也能用于本实现方式. 第一，可采用 FF-VF（具有空填充的最先适合）方法，其中，P_M 中的未调度时间的次序和 P_G 中的间隙并不按某一次序（下降或上升次序）排序，而是利用所找到的最先合格的数据信道来载送数据子帧. 第二，可采用 LAUC（最近有效未调度信道）方法，其中，未使用 P_G，即不考虑空填充. 这还可以简化设计. 第三，可采用 FF（最先适合）方法. FF 是 FF-VF 的一种简化形式，其中没有使用空填充.

图 9 中示出了 CCS 模块 86 的框图. 与 DCS 模块 84 类似，关联处理器 P_T 120 跟踪控制信道的使用. 由于每帧能发送最大 P_f 个字节的

有效载荷，因此，P_T 120 的存储器 T 121 只跟踪每帧可用的字节数（图 10）。这里，同样采用相对时间。CCS 模块 86 具有两个 b_1-bit 帧计数器 C^f 和 C^f_1。C^f 计数时帧。C^f_1 记录自从接收到最近的 BHP 所逝去的帧。一旦接收到到达时间为 t_{DB} 的 BHP，CP_2 122 就将接收到这一 BHP 的帧标上时间标记，即 $t^f_{sch} \leftarrow C^f$。同时，它读取计数器 C^f_1（$t^f_e \leftarrow C^f_1$），并将 C^f_1 复位到 0。然后，它通过将 B_i 下移 t^f_e 个位置更新 P_T，即 $B_{i-t^f_e} = B_i$（$t^f_e \leqslant i \leqslant 2^{b1}-1$），和 $B_i = P_f$（$2^{b1}-t^f_e \leqslant i \leqslant 2^{b1}-1$）。在初始化时，$P_T$ 中的所有条目都被置为 P_f。接着，CP_2 利用下式计算出数据子帧出发时（假定使用 FDL Q_i）的帧 t^f_{DB}：

$$t^f_{DB}(Q_i) = \left\lfloor ((t_{DB} + Q_i) \bmod 2^b)/T_f \right\rfloor, \quad 0 \le i \le B, \tag{6}$$

其中，T_f 是以时隙为单位的帧长度。根据下式计算出 DB 出发的相对时帧：

$$t'^f_{DB}(Q_i) = (t^f_{DB}(Q_i) - t^f_{sch} + 2^{b_1}) \bmod 2^{b_1}, \quad 0 \le i \le 2. \tag{7}$$

参数 b_1 可以根据参数 b 来估算，例如，$2^{b1}=2^b/T_f$。当 b=8 且 T_f=4 时，b_1=6。采用以下方法可以搜寻针对某一 DB 出发时间 t（例如，$t=t'^f_{DB}(Q_i)$）可能的 BHP 出发时间。基本思想是尽可能早地发送 BHP，但偏移时间不应大于τ_0（结合图 4a 和 4b 中所述）。设 J=「τ_0/T_f」。例如，当τ_0=8 个时隙而 T_f=1 个时隙时，J=8。当τ_0=8 个时隙而 T_f=4 个时隙时，J=2。假定，BHP 长度为 X 个字节。

在这一优选实施方式中，用强制最早时间（CET）方法来调度控制信道，如图 11 中所示。在步骤 130 中，P_T 120 将 X（即 BHP 的长度）与存储器 T 121 的相关条目 E_{t-j} 的内容 B_{t-j} 并行比较，$0 \leqslant j \leqslant J-1$，而 t-j>0。如果 $X \leqslant B_{t-j}$，那么条目 E_{t-j} 返回 1，否则，它返回 0。在步骤 132 中，如果 P_T 中至少一个条目返回 1，那么，在步骤 134 中选择最小指数的条目。保存该指数，并且，CCS 模块 86 报告已找到发送 BHP 的帧。如果，P_T 中没有条目返回“1”，那么，向 BHP 处理器 82 发送一个否认（步骤 136）。

如果选择 E_{t-j}，那么发出 BHP 的实际帧 t_f 为（$t'_{DB}-j+2^{b1}$）$mod2^{b1}$. 新的脉冲串偏移时间为（$t_{DB}modT_f$）$+j*T_f$.

在运用了 CET 方法后，CCS 模块 86 向 BHP 处理器 82 发送关于是否调度 BHP 以及在哪个时帧发送该 BHP 的信息. 一旦得到来自 BHP 处理器 82 的 ACK，CCS 模块 86 就将更新 P_T. 例如，如果条目 y 中的内容需要被更新，那么，$B_y \leftarrow B_y-X$. 如果无法调度 BHP，那么 CCS 模块 86 将向 BHP 处理器 82 发送一个 NACK. 在实际实现方式中，P_T 中的内容不必实际移动. 可用一个指针来记录与参考时帧 0 相关的条目指数.

为了并行调度，如下所述，由于 CCS 模块 86 不知道数据子帧的实际出发时间，因此，它可以在数据子帧的所有可能的出发时间或其子集调度 BHP，并向 BHP 处理器 82 报告结果. 当 B=2 时，有三个可能的数据子帧出发时间 t'_{DB}、$t'_{DB}+Q_1$ 和 $t'_{DB}+Q_2$. 象 DCS 模块 84 一样，如果使用三个相同的 P_T'，如图 9 中所示，那么，一个并行搜索将判断是否可以在这三个可能的数据子帧出发时间调度该 BHP.

图 12 中示出了路径和信道选择器 44 的框图. 路径和信道选择器 44 的功能在于，控制对 R × R RB 交换机 26 的接入，并指令 RB 交换控制器 28 和空分交换控制器 30 对各自的交换机 26 和 24 进行配置. 路径和信道选择器 44 包括处理器 140，它与循环缓冲输入调度（RBIS）模块 142、循环缓冲输出调度（RBOS）模块 144 和队列 146 连接. RBIS 模块 142 跟踪发向 RB 交换机 26 的 R 个入口信道的使用，而 RBOS 模块 144 跟踪发自 RB 交换机 26 的 R 个出口信道的使用. 在 RBIS 和 RBOS 模块 142 和 144 中，可以采用任意调度方法，例如，LAUC-VF、FF-VF、LAUC、FF 等. 注意，RBIS 模块 142 和 RBOS 模块 144 可以采用相同或者不同的调度方法. 从制作角度来看，RBIS 和 RBOS 模块最好采用与 DCS 模块 84 相同的调度方法. 在不失一般性的情况下，这里假定，在 RBIS 和 RBOS 模块 142 和 144 中均采用 LAUC-VF 方法；这样，DCS 模块的设计方案可以再利用，从而可应用于这些模块。

假定，时长为 l_{DB} 的数据子帧在时间 t_{DB} 到达 OSM 并且要求延时 Q_i。处理器 140 同时启动 RBIS 模块 142 和 RBOS 模块 144。它将 t_{DB} 和 l_{DB} 的信息发送给 RBIS 模块 142，并将离开 OSM 的时间（$t_{DB}+Q_i$）和 l_{DB} 的信息发送给 RBOS 模块 144。RBIS 模块 142 搜寻发向 RB 交换机 26 的在（t_{DB}，$t_{DB}+l_{DB}$）时间段内空闲的入口信道。如果有两个或两个以上合格的入口信道，那么 RBIS 模块将根据 LAUC-VF 选择一个信道。同样，RBOS 模块 144 搜寻发自 RB 交换机 26 的在（$t_{DB}+Q_i$，$t_{DB}+l_{DB}+Q_i$）时间段内空闲的出口信道。如果有两个或两个以上合格的出口信道，那么 RBOS 模块 144 将根据 LAUC-VF 选择一个信道。RBIS（RBOS）模块将所选择的入口（出口）信道标识符或 NACK 发送给处理器。如果找到了发向 RB 交换机 26 的合格的入口信道和发自 RB 交换机 26 的合格的出口信道，那么处理器将向 RBIS 和 RBOS 模块分别发回一个 ACK，于是，这两个模块将更新信道状态信息。同时，处理器还向调度器 42 发送一个 ACK，并向两个交换控制器 28 和 30 发送配置信息。否则，处理器 140 将向 RBIS 和 RBOS 模块 142 和 144 发送 NACK，并向调度器 42 发送一个 NACK。

需要 RBOS 模块 144 是因为，数据子帧所要使用的 FDL 缓冲器是由调度器 42 选择的，而不是由 RB 交换机 26 确定的。因此，数据子帧很可能可以进入 RB 交换机 26，但很可能由于出口信道冲突而无法从 RB 交换机 26 出去。图 13 中示出了一个例子，其中，三个固定长度的数据子帧 148a-c 到达 2×2 RB 交换机 26。前两个数据子帧 148a-b 将延时 2D，而第三个 DB 将被延时 D。显然，这三个数据子帧将同时离开交换机并争用两个出口信道。本例中，第三个数据子帧 148c 将丢失。

BHP 传输模块 46 负责在 BHP 处理器 82 所确定的时帧中在出口控制信道 52 上发送 BHP。由于在时隙传输中，帧有效载荷是固定的，都等于 P_p。因此，图 14 中示出了一种可能的实现方式，其中，整个存储器被划分成 W_c 段 150，并且要以同一时帧发送的 BHP 被保存在同一个段 150 中。W_c 是控制信道调度窗，它等于 2^{bl}。每段都有一个存

储指针（如段 W_0 中所示），用于指向新 BHP 可被保存的存储地址。为了区分一帧中的 BHP，帧额外开销应当包括一个指示帧中 BHP 个数的字段。再者，每个 BHP 都应当包括一个长度字段，用于指示 BHP 的第一个字节到最后一个字节的分组长度（例如，以字节为单位）。

假定，t_c 是 BHP 传输模块接收到 BHP 的当前时帧，而 p_c 指向当前存储段。给定 BHP 出发时帧 t_f，那么，根据 $(p_c+(t_f-t_c+2^{b1}) \bmod 2^{b1}) \bmod 2^{b1}$ 计算出保存这一 BHP 的存储段。

图 15 示出了采用无源 FDL 环路 160 作为循环缓冲器的光路由器体系结构，其中，循环信道的个数 $R=R_1+R_2+\ldots+R_B$，其第 j 个信道群引入 Q_j 延时，$1 \leqslant j \leqslant B$。这里，循环信道有区别，而在图 1b 中，所有循环信道等同，这里的循环信道可以提供 B 个不同的延迟。采用无源 FDL 环路的潜在问题在于访问共享 FDL 缓冲器的阻塞可能性较大。例如，假定 B=2，R=4 和 $R_1=2$，$R_2=2$，并且 R_1 的两个循环信道当前正在使用中。如果新 DB 要求被延时 Q_1，那么在图 1b 中可以顺利地调度该 DB，因为还有两个空闲的循环信道。然而，在图 15 中无法调度它，因为能够延时 Q_1 的两个信道忙。

SCU 32 的设计方案除了下列一些变动之外与前面所述情况几乎相同：(1)路径和信道选择器 44 中的 RBOS 模块 144（参见图 12）不再需要，(2)RBIS 模块 142 中需要略有修改，以便区分循环信道（如果 B>1 的话）。为了减小 B>1 时对 FDL 缓冲器的存取的阻塞可能性，要求调度器为需要被缓冲的每个数据子帧都提供一个以上的延迟选择。下面提出了对调度器以及路径和信道选择器 44 的设计方案的影响情况。在不失一般性的情况下，在下面的讨论中，假定调度器 42 必须针对 B+1 个可能的延迟来调度数据子帧和 BHP。

在本实现方式中，图 5 中所示的 DCS 模块 84 的设计方案仍有效。搜索结果可以按表 1（假定 B=2）中所示的格式被保存，表中，指示符（1/0）指示出对于给定的延迟（比如说 Q_i）是否找到合格的数据信道。存储器类型（0/1）指示出 P_M 或 P_G。条目指数给出存储器中的位置，它用于以后进行信息更新。信道标识符列给出所找到的信道的标

识符. 于是, DCS 模块将指示符列和信道标识符列（仅指示符为 1 的那些列）传送给 BHP 处理器.

表 1: DCS 模块中保存的搜索结果（B=2）

	指示符 (1 bit)	存储器类型 (1 bit)	条目指数 Max($\log_2$G, $\log_2$K) bits	信道标识符 ($\log_2$K bits)
Q_0				
Q_1				
Q_2				

图 9 中所示的 CCS 模块 86 的设计方案也仍有效. 搜索结果可以按表 2（假定 B=2）中所示的格式被保存, 表中, 指示符（1/0）指示出对于给定的 DB 出发时间在控制信道上是否能调度 BHP. 条目指数给出存储器中的位置, 它用于以后进行信息更新. "发送 BHP 的帧"列给出 BHP 被调度以便发出的时帧. 于是, CCS 模块将指示符列和"发送 BHP 的帧"列（仅指示符为 1 的那些列）传送给 BHP 处理器.

表 2: CCS 模块中保存的搜索结果（B=2）

	指示符 (1 bit)	条目指数 (b_1 bits)	发送 BHP 的帧 (b_1 bits)
Q_0			
Q_1			
Q_2			

在比较了 DCS 和 CCS 模块中的指示符列后, 图 3 中的 BHP 处理器 82 知道, 对于给定的 FDL 延迟 Q_i（$1 \leqslant i \leqslant B$）, 是否能调度数据子帧及其 BHP, 并判断将哪些配置信息发送给图 12 中路径和信道选择器 44. 三种可能的设想是: (1)未使用 FDL 缓冲器可调度数据子帧, (2)使用 FDL 缓冲器可调度数据子帧, 和(3)无法调度数据子帧.

在第三种情况下，只能丢弃数据子帧及其 BHP. 在第一种情况下，将向路径和信道选择器发送下列信息：入口 DCG 标识符，入口数据信道标识符，出口 DCG 标识符，出口数据信道标识符，数据子帧到空分交换机的到达时间，数据子帧时长，FDL 标识符 0（即 Q_0）. 路径和信道选择器 44 在接收到这些信息后立刻发回一个 ACK. 在第二种情况下，将向路径和信道选择器发送下列信息：

- 入口 DCG 标识符，
- 入口数据信道标识符，
- 候选 FDL 缓冲器的个数 x，
- for(i=1to x do)
  - 出口 DCG 标识符，
  - 出口数据信道标识符，
  - FDL 标识符 i
- 数据子帧到空分交换机的到达时间，
- 数据子帧时长

在第二种设想中，通信和信道选择器 44 将搜寻空闲缓冲信道以便载送数据子帧. RBIS 模块 142 类似于结合图 12 所述的模块，只是，在这里，对于每个延迟为 Q_i 的信道群，$0 \leqslant i \leqslant B$，它都有一个 P_M 和 P_G 对. 作为例子，图 16 中示出了 B=2 时的一个例子. 利用一个并行搜索，RBIS 模块将知道能否调度数据子帧. 当 x=1 时，RBIS 模块 142 根据 BHP 处理器 82 选择了哪个 FDL 缓冲器执行对（P_{M1} 90a，P_{G1} 92a）或（P_{M2} 90b，P_{G2} 92b）的并行搜索. 如果找到了空闲缓冲信道，那么它将通知处理器 140，处理器 140 又向 BHP 处理器 82 发送一个 ACK. 当 x=2 时，将搜索（P_{M1}，P_{G1}）和（P_{M2} 和 P_{G2}）. 如果找到了两个不同延迟的空闲信道，那么选择延迟为 Q_1 的信道. 在这种情况下，向 BHP 处理器 82 发送一个 ACK 以及选择 Q_1 的信息. 成功搜索后，RBIS 模块 142 将更新相应的 P_M 和 P_G 对.

图 17-26 示出了上面所引用的 LAUC-VF 方法的变形. 在上面所引用的 LAUC-VF 方法中，两个关联处理器 P_M 和 P_G 用来保存同一外

出链路的所有信道的状态. 具体地说, P_M 保存 r 个字, 其中每个字都对应外出链路的 r 个数据信道中的一个信道. 它用来记录这些信道的未调度时间. P_G 包含 n 个超字, 每个超字对应某个数据信道中的一个可用时间间隔（一个间隙）. 存储在 P_M 和 P_G 中的这些时间都是相对时间. P_M 和 P_G 支持关联搜索操作, 以及数据移动操作, 以便按排定的次序保存这些时间. 由于进行并行处理, P_M 和 P_G 用作主要部件, 以满足严格的实时信道调度要求.

在图 22-23 中所述的实施方式中, 用于同一外出链路的一对关联处理器 P_M 和 P_G 被合成一个关联处理器 P_{MG}. 用统一的 P_{MG} 取代一对 P_M 和 P_G 的好处是可以简化总的核心路由器实现方式. 根据 ASIC 实现方式, P_{MG} 的开发费用大大低于一对 P_M 和 P_G 的开发费用. P_{MG} 可用来实现 LAUC-VF 方法的一种更简单的运行更快的变形.

在图 17a 和 17b 中, 示出了两个外出信道 Ch_1 和 Ch_2, 其中 t_0 为当前时间. 就 t_0 而言, 信道 Ch_1 有两个所调度的 DB, 即 DB_1 和 DB_2, 而 Ch_2 有一个所调度的 DB_3. Ch_1 上的 DB_1 与 DB_2 之间的时间称为间隙, 它是未被任何 DB 占用的最大时间间隔. 标为 t_1 和 t_2 的时间分别是 Ch_1 和 Ch_2 未调度的时间. 在 t_1 和 t_2 之后, Ch_1 和 Ch_2 各自可用于发送任何 DB.

LAUC-VF 方法设法根据某些优先顺序调度 DB. 例如, 假定新的数据子帧 DB_4 在 t'时间到达. 针对图 17a 中的情况, 可以在 Ch_2 的未调度时间之后, 在 Ch_1 上或在 Ch_2 上的间隙内调度 DB_4. LAUC-VF 方法为 DB_4 选择了 Ch_1, 并且, 根据一个原来的间隙得到两个间隙. 针对图 17b 中的情况, DB_4 与 Ch_1 上的 DB_1 冲突, 又与 Ch_2 上的 DB_3 冲突. 但通过利用 FDL 缓冲器, 它可以在不与 Ch_1 上的 DB 和/或 Ch_2 上的 DB 冲突的情况下被调度后进行传输. 图 17b 示出了这样一种调度, 即 DB_4 被分配到 Ch_2 上, 并得到一个新的时隙.

假定, 外出链路具有 r 个数据信道, 那么, 这一链路的状态其特征在于以下两个集合:

$S_M=\{(t_i,i)|t_i$ 是信道 Ch_i 的未调度时间$\}$

$S_G=\{(l_j,r_j,c_j)|l_j<r_j$ 而间隔 $[l_j,r_j]$ 是信道 Ch_{cj} 上的间隙$\}$

在美国系列号 09/689,584 中所提出的 LAUC-VF 的实施方式中，提出两个关联处理器 P_M 和 P_G 分别用来表示 S_M 和 S_G. 由于存储器字长固定，保存在 P_M 的关联存储器 M 和 P_G 的关联存储器 G 中的时间都是相对时间. 假定，当前时间为 t_0. 于是，小于 t_0 的任何时间值不能用来调度新的 DB. 设:

$$S'_M=\{(\max\{t_i-t_0,0\},i)\mid(t_i,i)\in S_M\}$$

$$S'_G=\{(\max\{l_j-t_0,0\},\max\{l_j-t_0,0\},c_j)\mid(l_j,r_j,c_j)\in S_G\}$$

S'_M 和 S'_G 中的时间是相对于当前时间 t_0（作为参考点 0）的时间. 因此，P_M 中的 M 和 P_G 中的 G 实际上分别用来保存 S'_M 和 S'_G.

美国系列号 09/689,584 中所提出信道调度器假定 DB 具有任意长度. 一种可能性是采取一种时隙传输方式. 在这种方式中，以时隙为单位发送 DB，而 BHP 成组发送，每组都由一个时隙来载送. 时隙时钟 CLK_s 用来确定时隙界限. 时隙传输由 CLK_s 脉冲来起动. 因此，以 CLK_s 周期数为单位来表示相对时间. CLK_s 脉冲如图 18 中所示. 除了时钟 CLK_s 之外，还有另一个更密的时钟 CLK_f. CLK_s 的周期是 CLK_f 周期的倍数. 在图 18 所示的例子中，一个 CLK_s 周期等于 16 个 CLK_f 周期. 时钟 CLK_f 用来协调 CLK_s 的周期内所执行的操作.

在图 19a 和 19b 中，提供了对美国系列号 09/689,584 中所提出的 P_M 和 P_G 的硬件设计的改进，以适应时隙传输. 在 P_M 中，有一个 r 个字的关联存储器 M. M 中的每个字 M_i 本质上都是一个寄存器，并且它与减法器 200 相关联. 寄存器 MC 保存一个算子. 在图 19a 的实施方式中，保存在 MC 中的值是自从 M 中最近更新后逝去的时间. 将保存在 MC 中的值播送给所有的字 M_i，$1\leqslant i\leqslant r$. 每个字 M_i 都进行如下操作：如果 $M_j>MC$，则 $M_j\leftarrow M_j-MC$；否则，$M_i\leftarrow 0$. 这一操作用来更新 M 中所保存的相对时间. 如果 MC 保存了自从执行最近时间并行减法之后逝去的时间，那么，再进行这一操作，将这些时间更新为相

对于进行这一新的“并行减法”时间的时间．另一种操作是并行比较．在这一操作中，将保存在 MC 中的值播送给所有的字 M_i，$1 \leqslant i \leqslant r$．每个字 M_i 都进行如下操作：如果 $MC>M_i$，则 $MFLAG_i=1$；否则，$MFLAG_i=0$．优先级编码器将信号 $MFLAG_i$（$1 \leqslant i \leqslant r$）变换成地址．这一地址和与该地址相应的字分别被输出到 M 的地址和数据寄存器．这一操作用来寻找发送给定 DB 的信道．类似地，两个减法器用于 P_G 中的关联存储器 G 的字，每个对应一个子字。

图 19b 中所示的另一设计用来将 M 中的每个字 M_i 实现成并行装载的递减计数器．系统时隙时钟 CLK_s 的每个脉冲都将使该计数器减 1．当计数器到达 0 时，将停止计数，而一旦计数器被置为新的正值，就将恢复计数．假定，在时间 t_0 计数器的值为 t’，而在时间 $t_i>t_0$，计数器的值为 t”．那么，t”是 t’的相同时间（但是相对于 t_i 而言），即 $t''=\max\{t'-(t_1-t_0),0\}$．注意，相对于新参考点 t_1 的任何负时间（即，$t'-(t_1-t_0)<0$）不用于先行信道调度．与每个字 M_i 相关的是比较器 204．该比较器用于并行比较操作．同样，可利用具有两个关联比较器的两个递减计数器来实现 P_G 中 G 的字．

本系统具有一个 c-bit 的循环递增计数器 C_s．时隙时钟 CLK_s 的每个脉冲都将使 C_s 的值加 1．设 $t_{latency}(BHP_i)$是路由器接收到 BHP_i 的时间与信道调度器接收到 BHP_i 的时间之间的时间，它以 S'_G 周期数为单位．值 c 按下式来选择:

$$2^c > \left\lceil \frac{\max t_{latency}(BHP_i)}{MAX_s} \right\rceil$$

其中，MAXs 是 CLK_s 周期内 CLK_f 的周期数．当路由器接收到 BHP_i 时，用操作 $timestamp_{recv}(BHP_i) \leftarrow C_s$ 来为 BHP_i 标记时间．当路由器调度器接收到 BHP_i 时，再用 $timestamp_{sch}(BHP_i) \leftarrow C_s$ 来为 BHP_i 标记时间．设:

$$D_i = (timestamp_{recv}(BHP_i) + 2^c - timestamp_{sch}(BHP_i)) \bmod 2^c$$

于是，DB_i 到路由器的光交换矩阵的相对到达时间（用时隙时钟 CLK_s 表示）为 $T_i=\Delta+\tau_i+D_i$，其中 τ_i 是 BHP_i 与 DB_i 之间的偏移时间，而 D_i 是固定输入 FDL 时间。利用作为参考点的执行 $timestamp_{sch}(BHP_i)\leftarrow C_s$ 的时隙时间，以及保存在 P_M 和 P_G 中的相对时间，可以正确地调度 DB_i。

在 LAUC-VF 方法的硬件实现中，关联处理器 P_M 和 P_G 分别用来保存和处理 S'_M 和 S'_G。在任何时候，$S'_M=\{(t_i,i)|1\leqslant i\leqslant r\}$ 而 $S'_G=\{(l_j,r_j,c_j)|l_j\geqslant 0\}$。$S'_M$ 中的二元组 (t_i,i) 表示信道 Ch_i 上的未调度时间，而 S'_G 中的三元组 (l_j,r_j,c_j) 表示信道 Ch_{cj} 上的时间间隙（间隔）$[l_j,r_j]$。未调度时间 t_i 可认为是半无穷大间隙 $[t_i,\infty]$。因此，通过将这一半无穷大间隙包含到 S'_G 中，就不再需要 S'_M。

具体地说，设 $S''_M=\{(t_i,\infty,i)|(t_i,i)\in S'_M\}$，并定义 $S'_{MG}=S''_M\cup S'_G$。合成 P_M 和 P_G 的基本思想是通过修改 P_G 来构成 P_{MG}，这样，可用 P_{MG} 来处理 S'_{MG}。我们提出了关联处理器 P_{MG} 的结构，用于取代 P_M 和 P_G。P_{MG} 利用关联存储器 MG 来保存 S'_M 中的二元组和 S'_G 中的三元组。同 P_G 中的 G 一样，MG 中的每个字都有两个子字，当一个字用来保存 (l_j,r_j,c_j) 时，其第一个子字用于 l_j，而第二个子字用于 r_j。当 MG 中的字来保存 S'_M 中的二元组 (t_j,i) 时，其第一个子字用于 t_i，而第二个子字留着未用。前 r 个字预定给 S'_M，而剩余的字预定给 S'_G。前 r 个字按其第一个子字的非增加的次序保存，而剩余的字也按其第一个子字的非增加的次序保存。下面定义 P_{MG} 的新的操作。

下面，将概括 P_M 和 P_G 的结构和操作，并定义 P_{MG} 的结构和操作。P_{MG} 之间的不同包括所用的地址寄存器的个数、优先级编码器和所支持的操作。可以看到，与使用 P_M 和 P_G 的实现方式相比，P_{MG} 可以无需任何延迟用来实现 LAUC-VF 方法。

核心路由器的外出数据信道具有 r 个信道（波长），用于数据传输。这些信道用 $Ch_1,Ch_2,\ldots,Ch_r$ 来表示。设 $S=\{t_i|1\leqslant i\leqslant r\}$，其中，$t_i$ 是信道 Ch_i 的未调度时间。换言之，在 t_i 之后的任何时间，信道 Ch_i 都可用于传输。给定一个时间 T'，P_M 是一个关联处理器，用于快速搜索

$T''=\min\{t_i|t_i \geq T'\}$，其中 T'是给定时间．假定，$T''=t_j$，那么，信道 Ch_j 被认为是一个候选数据信道，用于在 T'时间发送 DB.

为了说明，在图 20 和 21 中示出了 P_M 和 P_C 的结构，而在图 22 中示出了 P_{MG}.

图 20 中示出了 P_M 210 的实施方式．关联处理器 P_M 包括一个关联存储器 M 212，该存储器有 k 个字 $M_1,M_2,\ldots,M_k$，每个字对应数据信道群的一个信道．每个字都与一个简单的用于减法和比较操作的减法电路相应．这些字还被连成一个线性数组．比较寄存器 MC 214 保存用于比较的算子．MCH 216 是一个存储器，该存储器有 k 个字 $MCH_1,MCH_2,\ldots,MCH_k$，其中 MCH_j 与 M_j 对应．这些字还被连成一个线性数组，并且它们用于保存信道编号．MAR_1 218 和 MAR_2 220 是地址寄存器，用于保存访问 M 和 MCH 的地址．MDR 222 和 MCHR 224 是数据寄存器，用于访问 M 和 MCHR 以及 MAR.

关联处理器 P_M 支持下列在有效实现 LAUC-VF 信道调度操作中所用的主要操作:

随机读：给出 MAR_1 中的地址 x，执行 $MDR_1 \leftarrow M_x$，$MCHR \leftarrow MCH_x$.

随机写：给出 MAR_1 中的地址 x，执行 $M_x \leftarrow MDR$，$MCH_x \leftarrow MCHR$.

并行搜索：将 MC 中的值与所有字 $M_1,M_2,\ldots,M_k$ 的值同时（并行）进行比较．找出最小的 j，使得 $M_j<MC$，并执行 $MAR_1 \leftarrow j$，$MDR_1 \leftarrow M_j$，和 $MCHR \leftarrow MCH_j$．如果所有的字 M_j 都不满足 $M_j<MC$，那么，在这一操作之后，$MAR_1=0$.

段下移：给出 MAR_1 中的地址 a，和 MAR_2 中的 b，使得 a<b，那么，对所有的 j（$a \leqslant j<b$），执行 $M_{j+1} \leftarrow M_j$ 和 $MCH_{j+1} \leftarrow MCH_j$.

对于“随机读”、“随机写”和“段下移”操作，每个二元组（M_j，MCH_j）都可以作为一个超字来处理．“并行搜索”的输出包括 r 个二进制信号 $MFLAG_i$（$1 \leqslant i \leqslant r$）．当且仅当 $M_i \leqslant MC$ 时，$MFLAG_i=1$. 有一个优先级编码器，其输入为 $MFLAG_i$（$1 \leqslant i \leqslant r$），当“并行搜索”

操作完成时，它将得到地址 j 并将该值装入 MAR_1 中。“随机读”、“随机写”、“并行搜索”和“段下移”操作用来保存 M 中所存储的值的非增加次序。

图 21 示出了关联处理器 P_G 92 的框图。P_G 用于保存核心路由器的外出链路的所有信道的未用间隙。间隙用一个整数二元组（l,r）来表示，其中 l 和 r 分别是间隙的起点和终点。关联处理器 P_G 包括关联存储器 G 93、比较寄存器 GC 230、存储器 GCH 232、地址寄存器 GAR 234、数据寄存器 GDR 236 和 GCHR 238 以及工作寄存器 GR_1 240 和 GR_2 242。

G 是一个关联存储器，它有 n 个字（$G_1,G_2,...,G_n$），每个 G_i 包括两个子字 $G_{i,1}$ 和 $G_{i,2}$。这些字被连成一个线性数组。GC 保存一个字，它有两个子字 GC_1 和 GC_2。GCH 是一个存储器，该存储器有 n 个字 $GCH_1,GCH_2,...,GCH_n$，其中 GCH_j 与 G_j 对应。这些字被连成一个线性数组，并且它们用于保存信道编号。GAR 是一个地址寄存器，用于保存访问 G 的地址。GDR 和 GCHR 是数据寄存器，用于访问 M 和 MCHR 以及 GAR。

关联处理器 P_G 支持下列在有效实现 LAUC-VF 信道调度操作中所用的主要操作：

随机写：给出 GAR 中的地址 x，执行 $G_{x,1} \leftarrow GDR_1$，$G_{x,2} \leftarrow GDR_2$，$GCH_x \leftarrow GCHR$。

并行双比较字搜索：将 GC 中的值与所有字 $G_1,G_2,...,G_n$ 的值同时（并行）进行比较。找出最小的 j，使得 $G_{j,1}<GC_1$ 和 $G_{j,2}>GC_2$。如果这一操作成功，那么，执行 $GDR_1 \leftarrow G_{j,1}$，$GDR_2 \leftarrow G_{j,2}$，$GCHR \leftarrow GCH_j$ 和 $GAR \leftarrow j$；否则，$GAR \leftarrow 0$。

并行单比较字搜索：将 GC_1 中的值与所有字 $G_1,G_2,...,G_n$ 的值同时（并行）进行比较。找出最小的 j，使得 $G_{j,1}>GC_1$ 并且 j 在寄存器 GAR 中。如果这一操作成功，那么，执行 $GDR_1 \leftarrow G_{j,1}$，$GDR_2 \leftarrow G_{j,2}$，$GCHR \leftarrow GCH_j$ 和 $GAR \leftarrow j$；否则 $GAR \leftarrow 0$。

双上移：给出 GAR 中的地址 a，那么，对 $a \leqslant j<n$，移位 G_{j+1} 的内容，$G_j \leftarrow G_{j+1}$，$GCH_j \leftarrow GCH_{j+1}$，$GCH_j$ 到 GCH_{j+1}，和 $G_{n,1} \leftarrow 0$，$G_{n,2} \leftarrow 0$.

双下移：给出 GAR 中的地址 a，执行 $G_{j+1} \leftarrow G_j$，$GCH_{j+1} \leftarrow GCH_j$，$a \leqslant j<n$.

在 P_G 中，三元组（$G_{i,1}$,$G_{i,2}$,GCH_i）对应于信道 GCH_i 上的一个间隙，其起始时间为 $G_{i,1}$ 而结束时间为 $G_{i,2}$. 对于“随机写”、“并行双比较字搜索”、“并行单比较字搜索”、“双上移”和“双下移”操作；每个三元组（$G_{i,1}$,$G_{i,2}$,GCH_i）都可以作为一个超字来处理。“并行双比较字搜索”（相应地，“并行单比较字搜索”）操作的输出包括 n 个二进制信号 $GFLAG_i$（$1 \leqslant i \leqslant n$），这样，当且仅当 $G_{i,1} \geqslant GC_1$ 且 $G_{i,2} \leqslant GC_2$（相应地，$G_{i,1} \geqslant GC_1$）时，$GFLAG_i=1$. 有一个优先级编码器，其输入为 $GFLAG_i$（$1 \leqslant i \leqslant n$），当该操作完成时，它将得到地址 j 并将该值装入 GAR_1 中。“随机写”、“并行单比较字搜索”、“双上移”和“双下移”操作用来保存 $G_{i,1}$ 中所存储的值的非增加次序.

美国系列号 09/689,584 中详细讨论了 P_M 和 P_G 的操作.

图 22 示出了处理器 P_{MG} 的框图，该处理器组合了上述 P_M 和 P_G 处理器的功能. P_{MG} 包括关联存储器 MG 248、比较寄存器 MGC 250、存储器 MGCH 252、地址寄存器 $MGAR_1$ 254a 和 $MGAR_2$ 254b、数据寄存器 MGDR 256 和 MGCHR 258.

MG 是一个关联存储器，它有 m=r+n 个字（MG_1,MG_2,...,MG_m），每个 MG_i 包括两个子字 $MG_{i,1}$ 和 $MG_{i,2}$. 这些字被连成一个线性数组。MGC 是一个比较寄存器，它保存一个字，该字有两个子字 MGC_1 和 MGC_2. MGCH 是一个存储器，该存储器有 m 个字 $MGCH_1$,$MGCH_2$,...,$MGCH_m$，其中 $MGCH_j$ 与 MG_j 对应. 这些字被连成一个线性数组，并且它们用于保存信道编号.

关联处理器 P_{MG} 支持下列主要操作：

随机读：给出 $MGAR_1$ 中的地址 x，执行 $MGDR_1 \leftarrow MG_{i,1}$，$MGDR_2 \leftarrow MG_{i,2}$，$MGCHR \leftarrow MGCH_x$.

随机写：给出 MGAR 中的地址 x，执行 $MG_{x,1} \leftarrow MGDR_1$，$MG_{x,2} \leftarrow MGDR_2$，$MGCH_x \leftarrow MGCHR$.

并行混合搜索：将 MGC_1 中的值与所有超字 MG_i（$1 \leqslant i \leqslant m$）的值并行进行比较，并将 MGC_1 和 MGC_2 中的值与所有超字 MG_j（$r+1 \leqslant j \leqslant m$）并行进行比较. (i) 如果 $MGC_2 \neq 0$，则并行执行下列操作：找出最小的 j'，使得，$j' \leqslant r$ 且 $MG_{j',1} < MGC_1$. 如果这一操作成功，那么，执行 $MGAR_1 \leftarrow j'$；否则 $MGAR_1 \leftarrow 0$. 找出最小的 j''，使得，$r+1 \leqslant j'' \leqslant m$，$MG_{j,1} < MGC_1$ 和 $MG_{j,2} > MGC_2$. 如果这一操作成功，那么，执行 $MGAR_2 \leftarrow j''$和 $MGCHR \leftarrow MGCH_i$；否则 $MGAR_2 \leftarrow 0$. (ii) 如果 $MGC_2=0$，那么，找出最小的 j'，使得，$1 \leqslant j' \leqslant m$ 且 $MG_{j',1} < MGC_1$，$MG_{j,2} > MGC_2$. 如果这一操作成功，那么，执行 $MGAR_1 \leftarrow j''$和 $MGCHR \leftarrow MGCH_i$；否则 $MGAR_1 \leftarrow 0$.

双上移：给出 $MGAR_1$ 中的地址 a，那么，对 $a \leqslant j < m$，执行 $MG_j \leftarrow MG_{j+1}$，$MGCH_j \leftarrow MGCH_{j+1}$，MGCH，到 $MGCH_{j+1}$，和 $MG_{n,1} \leftarrow 0$，$MG_{n,2} \leftarrow 0$.

段下移：给出 $MGAR_1$ 中的地址 a，和 $MGAR_2$ 中的 b，使得 a<b，那么，对所有的 j（$a \leqslant j < b$），执行 $MG_{j+1} \leftarrow MG_j$ 和 $MGCH_{j+1} \leftarrow MGCH_j$.

如 P_G 中那样，三元组（$MG_{i,1}$,$MG_{i,2}$,$MGCH_i$）可对应于信道 $MGCH_i$ 上的一个间隙，其起始时间为 $MG_{i,1}$ 而结束时间为 $MG_{i,2}$. 但在这种情况下，必须是 i>r。如果 $i \leqslant r$，那么，$MG_{i,2}$ 并不重要，二元组（$MG_{i,1}$,$MGCH_i$）被解释为信道 $MGCH_i$ 上的未调度时间 $MG_{i,1}$，并且这个二元组对应于 P_M 中的一个字. 对于“随机读”、“随机写”、“并行混合搜索”、“双上移”和“段下移”操作，每个三元组（$MG_{i,1}$,$MG_{i,2}$,...,$MGCH_i$）都可以作为一个超字来处理. 前 r 个超字用来保存 r 个外出信道的未调度时间，而后个 m-r 个超字用来保存关于所有外出信道上的间隙的信息.

“并行混合搜索”操作的输出包括一些二进制信号 $MGFLAG_i$，其值定义如下：(i) 如果 $MGC_2=0$ 且 $MG_{i,1} \geqslant MGC_1$，则 $MGFLAG_i=1$；(ii) 如果 $MGC_2 \neq 0$，$i \leqslant r$，$MG_{i,1} \geqslant MGC_1$，则 $MGFLAG_i=1$；(iii) 如果 $MGC_2 \neq 0$，$i>r$，$MG_{i,1} \geqslant MGC_1$，$MG_{i,2} \leqslant MGC_2$，则 $MGFLAG_i=1$，或者，如果 $MG_{i,1} \geqslant MGC_1$ 且 $i \leqslant r$，则 $MGFLAG_i=1$；和(iv) 否则，$MGFLAG_i=0$.

有两个编码器．第一个编码器其输入为 $MGFLAG_i$（$1 \leqslant i \leqslant r$），如果 $MGC_2 \neq 0$，那么，当“并行混合搜索”操作被执行后，它将在 $MGAR_1$ 中产生一个地址．第二个编码器其输入为 $MGFLAG_i$（$r+1 \leqslant i \leqslant m$）．如果 $MGC_2 \neq 0$，那么，当“并行混合搜索”操作被执行后，它将在 $MGAR_2$ 中产生一个地址．有一个选择器，以两个编码器的输出作为其输入．如果 $MGC_2=0$，那么，当“并行混合搜索”操作被执行后，将两个编码器产生的最小非零地址（如果有这种地址的话）装入到 $MGAR_1$ 中；否则，当“并行混合搜索”操作被执行后，将 $MGAR_1$ 置零；如果 $MGC_2 \neq 0$，那么，选择器的输出失能．

“随机读”、“随机写”、“并行混合搜索”、“双上移”和“段下移”操作用来保存 $MG_{i,1}$ 中前 m 个字的所存储的值的非增加次序，以及 $MG_{i,1}$ 中后 m-r 个字的所存储的值的非增加次序．

当关联处理器 P_M 和 P_G 用来实现 LAUC-VF 信道调度方法时，P_{MG} 的操作可以在没有任何延迟的情况下完成 P_M 和 P_G 操作．假定，P_{MG} 包括 $m=r+n$ 个超字．在表 3（相应地，表 4）中，右边一列中所给出的 P_{MG} 的操作用来完成左边一列中所给出的 P_M（相应地，P_G）的操作．不用同时搜索 P_M 和 P_G，而使用 P_{MG}，这一步骤可以由“并行混合搜索”操作来完成．

表 3: 用 P_{MG} 来模拟 P_M

P_M	P_{MG}
"随机读"	"随机读"
"随机写"	"随机写"
"并行搜索"	"并行混合搜索"
"段下移"	"段下移" （其中 $MGAR_2$=m）

表 4: 用 P_{MG} 来模拟 P_G

P_G	P_{MG}
"随机写"	"随机写"
"并行双比较字搜索"	"并行混合搜索"
"并行单比较字搜索"	"并行混合搜索" （其中 MGC_2=0）
"双上移"	"段上移" （其中 $MGAR_2$=m）
"双下移"	"段下移" （其中 $MGAR_2$=m-1）

在 LAUC-VF 方法中，最好使给定的 DB 适合间隙，即使 DB 可以在另一信道上被调度（在其未调度时间之后），如图 17a-b 中的例子所示．对于单独的 P_M 和 P_G，同时执行 P_M 和 P_G 的搜索操作，这一优先级是合理的．然而，执行这种操作的总电路可能太复杂．

通过将 P_M 和 P_G 合成一个关联处理器，这种 LAUC-VF 方法的既简单又快速的变形是可行的．图 23 中示出了另一种实施方式．该图中，处理器 P'_{MG} 270 包括一个数组 TYPE 272，它有 m 比特，每个比特与存储器 MG 中相应的字有关联．如果 $TYPE_i$=1，那么 MG_i 保存一项 S'_M，否则，MG_i 保存一项 S'_G．再者，寄存器 TYPER 274 是一个 1 比特寄存器，用于访问 TYPE 以及 $MGAR_1$ 和 $MGAR_2$．

P^{*}_{MG} 与 P_{MG} 之间的其他差别包括所用的优先级编码器和所支持的操作. 当调度新的 DB 时，搜索 MG. 所找到的合适时间间隔（不管它是一个间隙还是一个半无穷大间隔）用于新的 DB. 一旦 DB 被调度，又得到另一个间隙. 只要 MG 中有足够的空间，就将新的间隙保存在 MG 中. 当 MG 充满时，可以丢失一项 S'_G. 但是，必须强制保留 S'_M 的所有的项.

设，$t_s^{out}(DB_i)$和 $t_e^{out}(DB_i)$分别是在路由器的输出端发送 DB 的第一个时隙和最后一个时隙的发送时间. 那么:

$$t_s^{out}(DB_i) = T_i + L_j$$

和

$$t_e^{out}(DB_i) = T_i + L_j + length(DB_i),$$

其中，T_i 是前面所定义的相对到达时间，L_j 是交换矩阵中为 DB_i 所选择的 FDL 延时，而 $length(DB_2)$是 DB_i 的长度，它以时隙数为单位. 假定在 DB 交换矩阵中有 q+1 个 FDL，$L_0,L_1,\ldots,L_q$，且 $L_0=0<L_1<L_2<\ldots<L_{q-1}<L_q$. LAUC-VF 的新的变形拟定如下:

方法: "信道调度"

开始

success ← 0;

对于 j=0 至 q，循环执行

$MGC_1 \leftarrow T_i+L_j$

$MGC_2 \leftarrow T_i+L_j+length(DB_i)$;

利用 P^{*}_{MG} 执行"并行混合搜索";

如果 $MGAR_1 \neq 0$，则

如果 $MGAR_1 \neq 0$，则

开始

输出 MGCHR，作为用于发送 DB_i 的信道编号

输出 L_j，作为所选择的用于 DB_i 的 FDL 延时;

利用 MGC_1 和 MGC_2 中的值更新 P'_{MG} 中的 MG,

success ← 1;

*退出/*退出这一循环*/*

结束

结束循环

*如果 success=0, 那么丢弃 DB_i/*调度 DB_i 失败*/*

结束

一旦调度了 DB，就更新 MG. 当一个新的间隙要加入到 MG，而 $TYPE_m$=1 时，忽略这一新的间隙。这样确保了没有属于 S'_M 的项丢失。

关联处理器 P^*_{MG} 支持下列主要操作:

随机读: 给出 $MGAR_1$ 中的地址 x，执行 $MGDR_1 \leftarrow MG_{i,1}$，$MGDR_2 \leftarrow MG_{i,2}$，$MCHR \leftarrow MGCH_x$，$TYPER \leftarrow TYPE_x$.

随机写: 给出 $MGAR_1$ 中的地址 x，执行 $MG_{x,1} \leftarrow MGDR_1$，$MG_{x,2} \leftarrow MGDR_2$，$MGCH_x \leftarrow MGCHR$，$TYPE_x \leftarrow TYPER$.

并行混合搜索: 将 MGC_1 中的值与所有超字 MG_i（$1 \leqslant i \leqslant m$）的值并行进行比较，并将 MGC_2 与所有超字 MG_i（$1 \leqslant i \leqslant m$，其 $TYPE_i$=0）并行进行比较。找出最小的 j'，使得，$TYPE_{j'}$=1 且 $MG_{j',1} < MGC_1$，或者，$TYPE_j$=0，$MG_{j,1} < MGC_1$ 和 $MG_{j,2} > MGC_2$。如果这一搜索成功，那么，执行 $MGAR_1 \leftarrow j'$，$TYPER \leftarrow TYPE_{j'}$，$MGCH \leftarrow MGCH_{j'}$；否则 $MGAR_1 \leftarrow 0$.

双上移，段下移: 与 P_{MG} 中相同。

操作中，$TYPE_i$ 的值表示 MG_i 中所保存的信息的类型。如 P_G 中那样，三元组（$MG_{i,1}$,$MG_{i,2}$,$MGCH_i$）可对应于信道 $MGCH_i$ 上的一个间隙，其起始时间为 $MG_{i,1}$ 而结束时间为 $MG_{i,2}$。但在这种情况下，必须是 $TYPE_i$=0。如果 $TYPE_i$=1，那么，$MG_{i,2}$ 并不重要，二元组（$MG_{i,1}$,$MGCH_i$）被解释为信道 $MGCH_i$ 上的未调度时间 $MG_{i,1}$，并且这个二元组对应于 P_M 中的一个字。对于“随机读”、“随机写”、“并

行混合搜索”、“双上移”和“段下移”操作，每个四元组（$MG_{i,1}$,$MG_{i,2}$, $TYPE_i$,$MGCH_i$）都可以作为一个超字来处理.

“并行混合搜索”操作的输出包括一些二进制信号 $MGFLAG_i$，其值定义如下：如果 $MGC_2 \neq 0$，TYPE=0，$G_{i,1} \geqslant GC_1$ 且 $G_{i2} \leq GC_2$，则 $MGFLAG_i$=1. 如果 MGC_2=0，且 $G_{i,1} \geqslant GC_1$，则 $MGFLAG_i$=1. 否则，$MGFLAG_i$=0. 有一个优先级编码器. 如果其输入为 $MGFLAG_i$（$1 \leqslant i \leqslant m$），那么，当“并行混合搜索”操作被执行后，它将在 $MGAR_1$ 中产生一个地址.

“随机读”、“随机写”、“并行混合搜索”、“双上移”和“段下移”操作用来保存 $MG_{i,1}$ 中所存储的值的非增加次序.

图 24 示出了利用多个关联处理器来快速调度. OBS 核心路由器的信道调度有严格的时间要求，因此，建议使用多个关联处理器（图 24 中用 P^*_{MG} 处理器 270 表示，这些处理器都是并行处理器）来实现调度方法. 假定在 DB 交换矩阵中有 q+1 个 FDL，L_0=0,L_1,...,L_q，且 $L_0<L_1<...<L_q$. 这些 FDL 在需要时使用，以延迟 DB，以便提高能成功调度 DB 的可能性. 在美国系列号 09/689,584 中所提出的 LAUC-VF 方法的实现方式中，利用不同的 FDL 反复搜索相同的一对 P_M 和 P_G，直到得到调度办法或者用尽了所有 FDL. 上述的“信道调度”方法采用了相同的思想.

为了提高调度速度，调度器 42 可以使用 q+1 个 P_M/P_G 对，每个 P_M/P_G 对用于一个 L_i. 在任何时间，所有 q+1 个 M 都有相同的内容，所有 q+1 个 MCH 都有相同的内容，所有 q+1 个 G 都有相同的内容，以及所有 q+1 个 GCH 都有相同的内容. 于是，可以在这些 P_M/P_G 对上同时执行为所有不同的 FDL 寻找调度办法. 至多一个搜索结果用于一个 DB. 同时用同样的锁步操作来更新所有 P_M/P_G 对，以确保它们保存同样的信息. 同样，可以用 q+1 个 P_{MG} 或 P^*_{MG} 来提高调度速度.

在图 24 中，多处理器系统 300 利用 q+1 个 P^*_{MG} 270 来实现上述的“信道调度”方法. 同样，可以利用多个 P_M/P_G 对或多个 P_{MG} 以类似的方式来实现 LAUC-VF 方法，以得到更好的性能. 多个 P^*_{MG} 270

包括 q+1 个关联存储器 $MG^0,MG^1,...,MG^q$. 每个 MG^j 都有 m 个字 $MG^j_1,MG^j_2,...,MG^j_m$, 每个 MG^j_i 都包括两个子字 $MG^j_{i,1}$ 和 $MG^j_{i,2}$. 有 q+1 个比较寄存器 $MGC^0,MGC^1,...,MGC^q$. 每个 MGC^j 都保存一个字, 该字有两个子字 MGC^j_1 和 MGC^j_2. MGCH: 有 q+1 个关联存储器 $MGCH^0,MGCH^1,...,MGCH^q$. 每个 $MGCH^j$ 都有 m 个字 $MGCH^j_1,MGCH^j_2,...,MGCH^j_m$. $MGCH^j$ 中的字被连成一个线性数组. 有 q+1 个线性数组 $TYPE^0,TYPE^1,...,TYPE^q$, 其中, $TYPE^j$ 有 m 个比特 $TYPE^j_1,TYPE^j_2,...,TYPE^j_m$. $MGAR_1$、MGAR 是地址寄存器, 用于保存访问 MG 和 MGCH 的地址. MGDR、TYPER、MGCH 是数据寄存器, 用于访问 MG、TYPE 和 MGCHR.

多处理器系统 300 支持下列主要操作:

随机读: 给出 $MGAR_1$ 中的地址 x, 执行 $MGDR_1 \leftarrow MG^0_{i,1}$, $MGDR_2 \leftarrow MG^0_{i,2}$, $MCHR \leftarrow MGCH^0_x$, $TYPER \leftarrow TYPE^0_x$.

随机写: 给出 $MGAR_1$ 中的地址 x, 执行 $MG^j_{x,1} \leftarrow MGDR_1$, $MG^j_{x,2} \leftarrow MGDR_2$, $MGCH^j_x \leftarrow MGCHR$, $TYPE^j_x \leftarrow TYPER$, $0 \leqslant j \leqslant q$.

并行混合搜索: 对于 $0 \leqslant j \leqslant q$, 将 MGC^j_1 中的值与所有超字 MG^j_1 ($1 \leqslant i \leqslant m$) 的值并行进行比较, 并将 MGC^j_2 与所有超字 MG^j_i ($1 \leqslant i \leqslant m$, 其 $TYPE^j_i=0$) 并行进行比较. 对于 $0 \leqslant j \leqslant q$, 找出最小的 k_j, 使得, $TYPE^j_{kj,1}=1$ 且 $MG^j_{kj,1}<MGC^j_1$, 或者, $TYPE^j_{kj}=0$, $MG^j_{kj,1}<MGC^j_1$ 和 $MG^j_{kj,2}>MGC^j_2$. 如果这一搜索成功, 设 $l_j=1$; 否则, 设 $l_j=0$. 找出 $FD=\min\{j|l_j=1,0 \leqslant j \leqslant q|\}$. 如果这样的 1 存在, 那么, 执行 $j \leftarrow FD$, $MGAR_1 \leftarrow k_j$, $TYPER \leftarrow TYPE^j_{kj}$, $MGCH \leftarrow MGCH^j_{kj}$; 否则 $MGAR_1 \leftarrow 0$.

双上移: 给出 $MGAR_1$ 中的地址 a, 那么, 对于 $0 \leqslant j \leqslant q$, 执行 $MG^j_i \leftarrow MG^j_{i+1}$, $MGCH^j_i \leftarrow MGCH^j_{i+1}$, $MGCH^j_i$ 到 $MGCH^j_{i+1}$, 对于 $a \leqslant i<m$, 和 $MG^j_{n,1} \leftarrow 0$, $MG^j_{n,2} \leftarrow 0$.

段下移: 给出 $MGAR_1$ 中的地址 a, 和 $MGAR_2$ 中的 b, 使得 $a<b$, 那么, 对于 $0 \leqslant j \leqslant q$, 执行 $MG^j_i \leftarrow MG^j_i$ 和 $MGCH^j_{i+1} \leftarrow MGCH^j_i$ (所有 $a \leqslant j<b$).

“随机读”操作在 P^*_{MG} 的一个副本即（MG^0,$TYPE^0$ 和 MGC^j）中执行. “随机写”、“并行混合搜索”、“双上移”和“段下移”操作在 P^*_{MG} 的所有副本中执行. 对于“随机读”、“随机写”、“并行混合搜索”、“双上移”和“段下移”操作，每个四元组（$MG_{i,1}$,$MG_{i,2}$, $TYPE_i$,$MGCH_i$）都可以作为一个超字来处理. 当“并行混合搜索”操作完成时，所有的 P^*_{MG} 副本的输出作为选择器的输入. 选择一个 P^*_{MG} 副本的输出.

“信道调度”方法可以在多处理器系统中按如下方法来实现:

方法: “并行信道调度”

开始

success ← 0;

对于 j=0 至 q，循环并行执行

$MGC^j_1 \leftarrow T_i+L_j$

$MGC^j_2 \leftarrow T_i+L_j+length(DB_i)$;

结束循环

执行“并行混合搜索”;

如果 $MGAR_1 \neq 0$，则

开始

输出 MGCHR，作为用于发送 DB_2 的信道编号;

输出 L，作为所选择的用于 DB_i 的 FDL 延时

$k \leqslant FD$;

对于 j=0 至 q，循环并行执行

利用 $MGCDR^k_1$ 和 $MGDR^k_2$ 中的值更新 MG^j（$0 \leqslant j \leqslant q$）

结束循环

success ← 1;

结束

*如果 success=0，那么丢弃 DB_i/*调度 DB_i 失败*/*

结束

可能希望能将 r 个数据信道划分成一些群并选择一个特殊群来调度 DB. 这种情况可能会出现在某些场合. 例如，我们想试验某个特殊信道. 在这种情况下，本身要试验的信道构成一个信道群，而其余所有信道构成另一个信道群. 那么，信道调度只在 1 个信道群上进行. 另一些情形是，在路由器的运行期间，某些信道可能不能发送 DB. 那么，同一外出链路的信道可划分为两个群，其中一个群包括所有失效信道，而另一个群包括所有正常信道，并且只能选用正常的信道来发送 DB. 划分数据信道还允许信道预留，它可应用于服务质量. 利用预留的信道群，可以构成虚拟电路和虚拟网络.

为了在信道调度关联处理器中引用群划分特性，基本思想是使群标识符（或简称“gid”）与各信道相关联. 对于链接，具有相同 gid 的所有信道都属于同一群. 信道的 gid 是可编程的；也就是说，可以根据需要动态地改变信道的 gid. 用于 DB 的 gid 可以从其 BHP 和/或某些其他本地信息中得到.

将 P_M 和 P_G 的设计方案扩展为 $P_{M\text{-}ext}$ 和 $P_{G\text{-}ext}$，可以应用于多个信道群，分别如图 25 和 26 中所示. 如图 25 中所示，关联处理器 $P_{M\text{-}ext}$ 290 包括 M、MC、MCH、MAR_1、MAR_2、MDR、MCHR，如参照图 20 所述. MCIDC 292 是一个比较寄存器，它保存用于比较的 gid. MGID 294 是一个存储器，该存储器有 r 个字 $MGID_1,MGID_2,\ldots,MGID_r$，其中 $MGID_j$ 与 M_j 和 MCH_j 对应. 这些字被连成一个线性数组，并且它们用于保存信道群编号. MGIDDR 296 是一个数据寄存器.

$P_{M\text{-}ext}$ 类似于 P_M，只是增加了几个部件，并且对操作进行了修改. 线性数组 MGID 有 r 个单元 $MGID_1,MGID_2,\ldots,MGID_r$；每个字都用来保存一个整数 gid. $MGID_i$ 与 M_i 和 MCH_i 对应，即三元组（M_i,MCH_i,$MGID_i$）可以作为一个超字来处理. 此外，增加比较寄存器 MGIDC 和数据寄存器 MGIDDR.

关联处理器 $P_{M\text{-}ext}$ 支持下列在有效实现 LAUC-VF 信道调度操作中所用的主要操作:

随机读：给出 MAR_1 中的地址 x，执行 $MDR \leftarrow M_x$，$MCH_x \leftarrow MCHR$ 和 $GIDR \leftarrow MGID_x$.

随机写：给出 MAR_1 中的地址 x，执行 $M_x \leftarrow MDR$，$MCH_x \leftarrow MCHR$ 和 $MGID_i \leftarrow MGIDDR$.

并行搜索 1：同时将 MGIDC 与 $MGID_1, MGID_2, \ldots, MGID_r$ 的值进行比较．找出 j，使得 MGID=MGIDC，并执行 $MAR_1 \leftarrow j$，$MDR_1 \leftarrow M_j$，$MCHR \leftarrow MCH_j$，和 $MGIDDR \leftarrow MGID+j$.

并行搜索 2：同时将(MC,MGIDC)与 $(M_1,MGID_1), (M_2,MGID_2), \ldots, (M_r,MGID_r)$进行比较．找出最小的 j，使得，$M_j<MC$ 且 $MGID_j=GIDC$，并执行 $MAR_1 \leftarrow j$，$MDR_1 \leftarrow M_j$，$MCHR \leftarrow MCH_j$，和 $MGIDDR \leftarrow MGID_1$．如果所有的字（$M_j,MGID_j$）都不满足 $M_j<MC$ 和 $MGID_j=GDIC$，那么，在这一操作之后，$MAR_1=0$.

段下移：给出 MAR_1 中的地址 a，和 MAR_2 中的 b，使得 a<b，那么，对所有的 j($a \leqslant j<b$)，执行 $M_{j+1} \leftarrow M_j$，$MCH_{j+1} \leftarrow MCH_j$ 和 $MGID_{j+1} \leftarrow MGID_j$.

对于“随机读”、“随机写”和“段下移”操作，每个三元组（$M_j,MCH_j,MGID_j$）都可以作为一个超字来处理．“并行搜索 1”的输出包括 r 个二进制信号 $MFLAG_i$（$1 \leqslant i \leqslant r$）．当且仅当 $MGID_i=MGIDC$ 时，$MFLAG_i=1$．有一个优先级编码器，其输入为 $MFLAG_i$（$1 \leqslant i \leqslant r$），当“并行搜索 1”操作完成时，它将得到地址 j 并将该值装入 MAR_1 中．“并行搜索 2”的输出包括 r 个二进制信号 $MFLAG_i$（$1 \leqslant i \leqslant r$）．当且仅当 $M_i \leqslant MC$ 和 $MGID_i=MGIDC$ 时，$MFLAG_i=1$．当“并行搜索”操作完成时，与“并行搜索 1”中所用相同的一个优先级编码器将 $MFLAG_i$（$1 \leqslant i \leqslant r$）变换到地址 j，并将该值装入 MAR_1 中．“随机读”、“随机写”、“并行搜索 2”和“段下移”操作用来保存 M 中所存储的值的非增加次序．

图 26 示出了 $P_{G\text{-}ext}$ 的框图．$P_{G\text{-}ext}$ 300 包括 G、GC、GCH、GAR、GDR、GCHR，如参照图 21 所述．GGIDC 302 是一个比较寄存器，它保存用于比较的 gid．GGID 304 是一个存储器，该存储器有 r 个字

$GGID_1,GGID_2,...,GGID_r$，其中 $GGID_j$ 与 G_j 和 GCH_j 对应．这些字被连成一个线性数组，并且它们用于保存信道群编号．GGIDR 306 是一个数据寄存器．

与 $P_{M\text{-}ext}$ 的结构类似，只是在 P_G 中增加了一个线性数组 GGID，该线性数组有 n 个字 $GGID_1,GGID_2,...,GGID_n$．四元组（$G_{i,1},G_{i,2},MCH_i,GGID_i$）可以作为一个超字来处理．

关联处理器 $P_{G\text{-}ext}$ 支持下列在有效实现 LAUC-VF 信道调度操作中所用的主要操作:

随机写：给出 GAR 中的地址 x，执行 $G_{x,1}\leftarrow GDR_1$，$G_{x,2}\leftarrow GDR_2$，$GCH_x\leftarrow GCHR$，$GGID_x\leftarrow GGIDR$，．

并行双比较字搜索：将（GC,GGIDC）中的值与$(G_1,GGID_1)$，$(G_2,GGID_2),...,(G_n,GGID_n)$同时（并行）进行比较．找出最小的 j，使得 $G_{j,1}<GC_1$，$G_{j,2}>GC_2$ 和 $GGID_j=GGIDC$．如果这一操作成功，那么，执行 $GDR_1\leftarrow G_{j,1}$，$GDR_2\leftarrow G_{j,2}$，$GCHR\leftarrow GCH_j$，$GGIDR\leftarrow GGID_j$ 和 $GAR\leftarrow j$；否则，$GAR\leftarrow 0$.

并行单比较字搜索：将$(GC_1,GGIDC)$与$(G_{1,1},GGID_1)$，$(G_{2,1},GGID_2),...,(G_{n,1},GGID_n)$同时（并行）进行比较．找出最小的 j，使得 $G_{j,1}>GC_1$ 和 $GGID_j=GGIDC$．如果这一操作成功，那么，执行 $GDR_1\leftarrow G_{j,1}$，$GDR_2\leftarrow G_{j,2}$，$GCHR\leftarrow GCH_j$，$GGIDR\leftarrow GGID$ 和 $GAR\leftarrow j$；否则 $GAR\leftarrow 0$.

双上移：给出 GAR 中的地址 a，那么，对 $a\leqslant j\leqslant n$，移位 G_{j+1} 的内容，$G_j\leftarrow G_{j+1}$，$GCH_j\leftarrow GCH_{j+1}$，$GCH_j$ 到 GCH_{j+1}，GGID 到 $GGID_{j+1}$，和 $G_{n,1}\leftarrow 0$，$G_{n,2}\leftarrow 0$.

双下移：给出 GAR 中的地址 a，执行 $G_{j+1}\leftarrow G_j$，$GCH_{j+1}\leftarrow CCH_j$，$GGID_{j=1}\leftarrow GCID_j$，$a\leqslant j<n$.

四元组$(G_{i,1},G_{i,2},GCH_i,GGID_i)$对应于信道 CCH_i（其 gid 在 $GGID_i$ 中）上的一个间隙，其起始时间为 $G_{i,1}$ 而结束时间为 $G_{i,2}$．对于“随机写”、“并行双比较字搜索”、“并行单比较字搜索”、“双上移”和“双下移”操作，每个四元组（$G_{i,1},G_{i,2},GCH_i,GGID_i$）都可以作为

一个超字来处理。“并行双比较字搜索”（相应地，“并行单比较字搜索”）操作的输出包括 n 个二进制信号 $GFLAG_i$（$1 \leqslant i \leqslant n$），这样，当且仅当 $G_{i,1} \geqslant GC_1$ 且 $G_{i,2} \leqslant GC_2$（相应地，$G_{i,1} \geqslant GC_1$），$GGID_i$=GGIDC 时，$GFLAG_i$=1. 有一个优先级编码器，其输入为 $GFLAG_i$（$1 \leqslant i \leqslant n$），当该操作完成时，它将得到地址 j 并将该值装入 GAR 中。“随机写”、“并行单比较字搜索”、“双上移”和“双下移”操作用来保存 $G_{i,1}$ 中所存储的值的非增加次序。

将信道 Ch_j 的 gid 从 g1 改变为 g2 按下列步骤进行：找出三元组（M_i,MCH_i,$MGID_i$），使得，MCH_i=j，并将 i 存入到 MAR_1 和（MDR，MCHR，MGIDDR）中；MGIDDR←g2，并将利用 MAR_1 中的地址 i 写回（MDR，MCHR，MGIDDR）。

给定一个 DB'、t_s^{out}(DB')、t_e^{out}(DB')和一个 gid g，那么 DB'的调度包括 $P_{M\text{-}ext}$ 和 $P_{G\text{-}ext}$ 中的搜索。$P_{M\text{-}ext}$ 中的搜索这样实现：找出一个最小的 i，使得，M_i<t_s^{out}(DB')和 $MGID_i$=g。$P_{G\text{-}ext}$ 中的搜索这样实现：找出一个最小的 i，使得，$G_{i,1}$<t_s^{out}(DB')，$G_{i,2}$>t_s^{out}(DB')，和 $MGID_i$=g。

同样，通过增加一个 gid 比较寄存器 MGGIDC、一个存储器 MGGID（有 m 个字 $MGGID_1$,$MGGID_2$,...,$MGGID_m$）和一个数据寄存器 MGGIDDR 可以构成关联处理器 $P_{G\text{-}ext}$ 和 $P_G{}^*{}_{\text{-}ext}$。$P_{MG\text{-}ext}$ 是 $P_{M\text{-}ext}$ 和 $P_{G\text{-}ext}$ 的合成。从 $P_{M\text{-}ext}$ 和 $P_{G\text{-}ext}$ 的操作可以容易地得出 $P_{MG\text{-}ext}$ 的操作，因为 $P_{M\text{-}ext}$ 条目和 $P_{G\text{-}ext}$ 条目是独立的。在 $P^*_{MG\text{-}ext}$ 中，混合了 $P_{M\text{-}ext}$ 条目和 $P_{G\text{-}ext}$ 条目。由于这些条目的 $MG_{i,1}$ 值是非增加次序的，因此，通过找到最小的 j，使得 $MGGID_j$=i 可以找到与信道 Ch_i 相应的 $P_{M\text{-}ext}$ 条目。

尽管以上针对几例实施方式详述了本发明，然而，对熟练技术人员而言，可以提出这些实施方式的各种修改方式以及其他实施方式。本发明包括了属于权利要求书范围的任何修改方式或其他实施方式。

图 1a

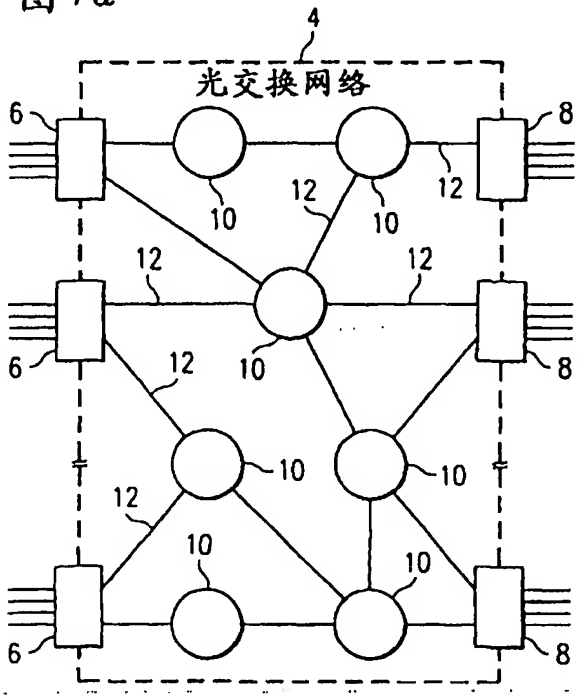
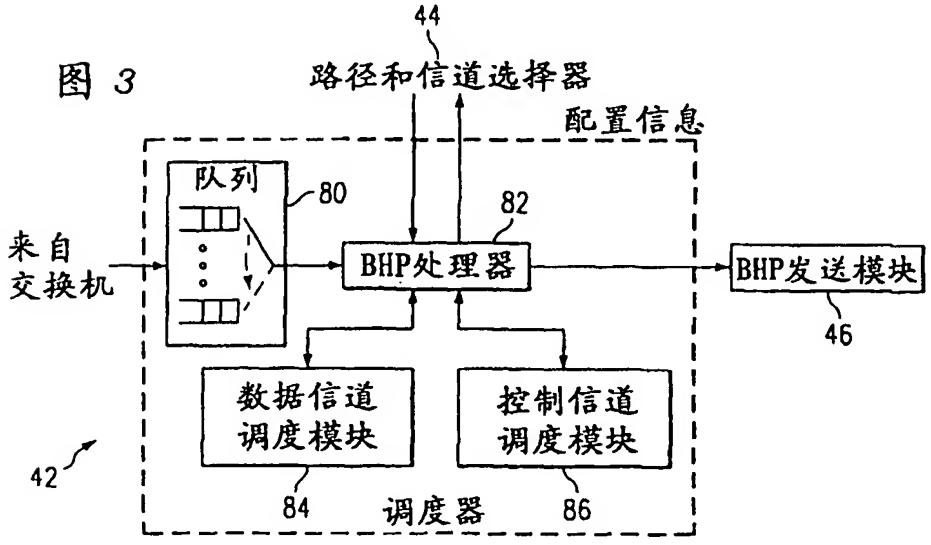

4
光交换网络
6
8
10
12
12
10
12
12
6
8
12
10
10
12
10
10
6
8

图 3


44
路径和信道选择器
配置信息
队列
80
82
来自
交换机
BHP处理器
BHP发送模块
46
数据信道
调度模块
控制信道
调度模块
42
84
调度器
86

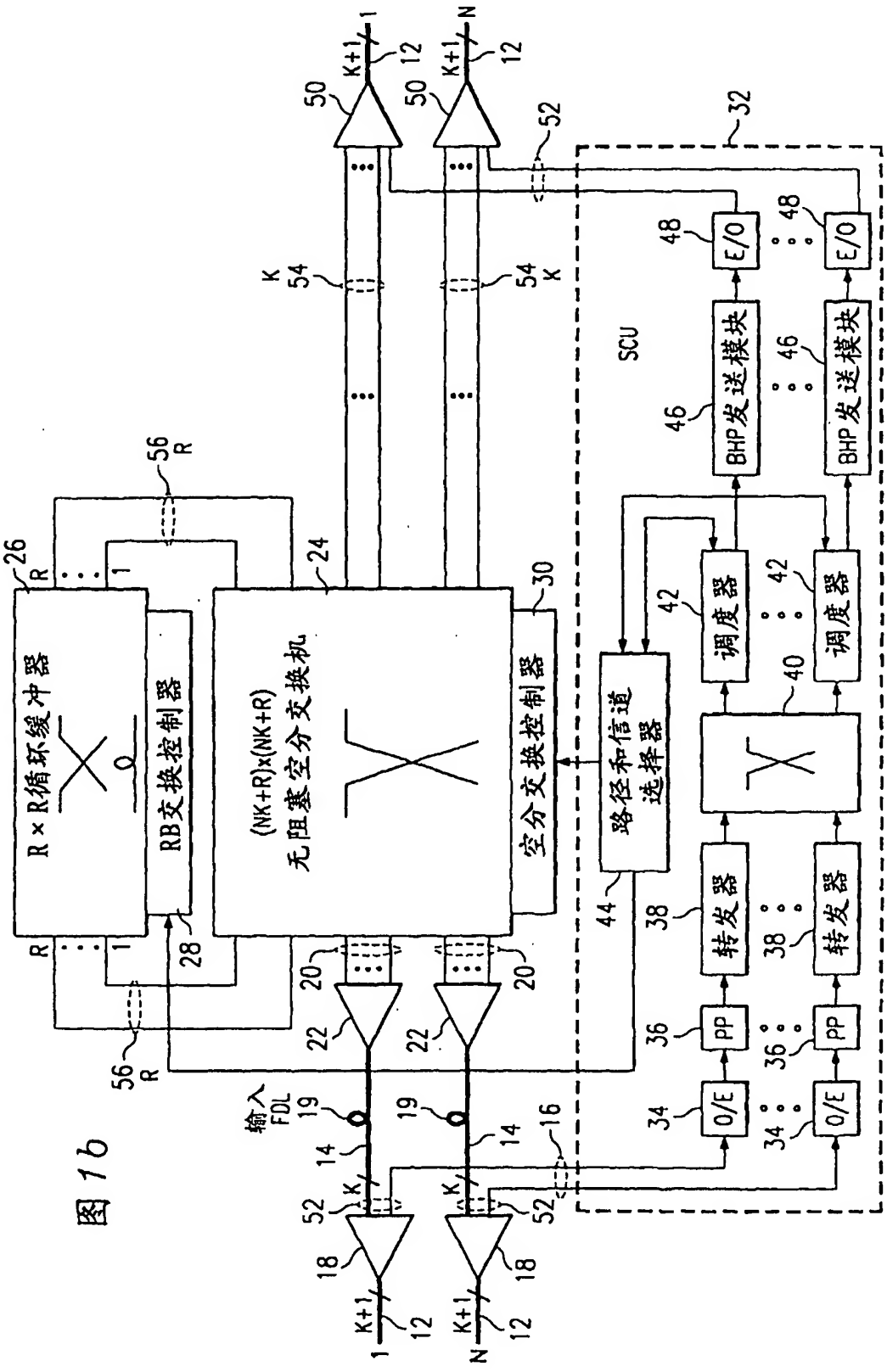
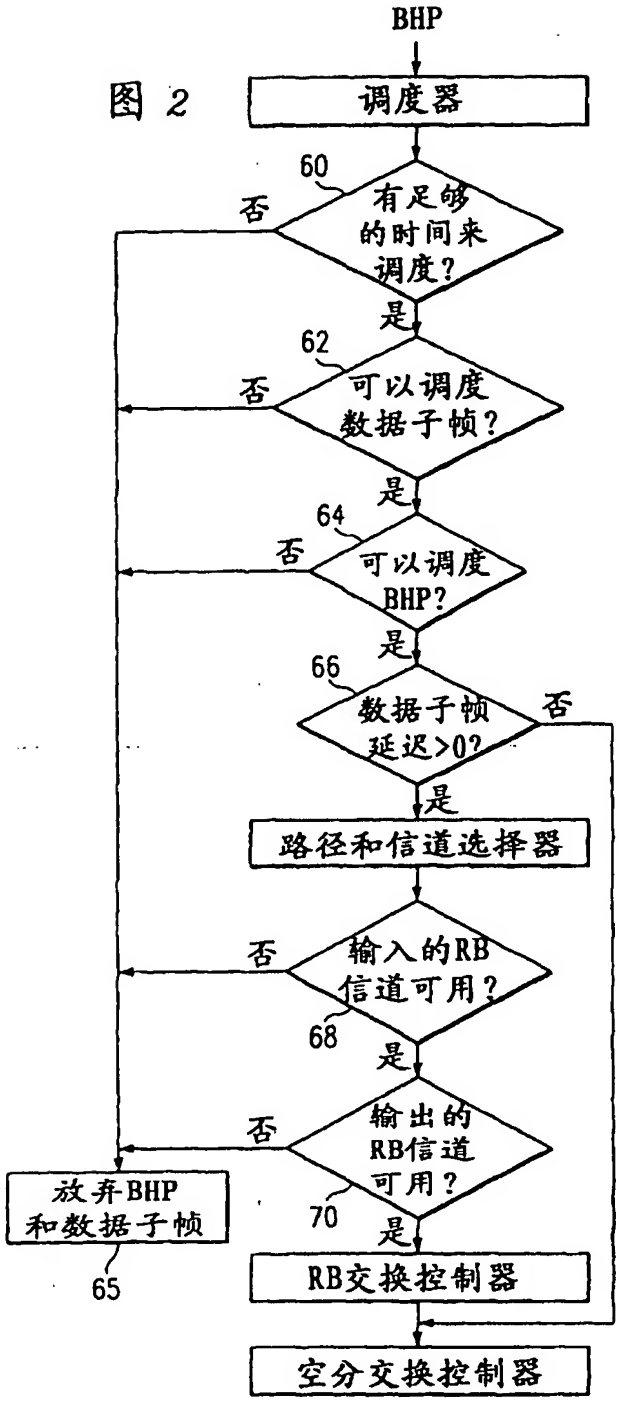

R × R循环缓冲器
RB交换控制器
(NK+R)x(NK+R)
无阻塞空分交换机
空分交换控制器
路径和信道
选择器
调度器
转发器
O/E
PP
E/O
BHP发送模块
SCU
输入
FDL

图 1b

图 2


BHP
调度器
60
有足够的时间来调度?
否
是
62
可以调度数据子帧?
否
是
64
可以调度BHP?
否
是
66
数据子帧延迟>0?
否
是
路径和信道选择器
输入的RB信道可用?
否
68
是
输出的RB信道可用?
否
70
是
放弃BHP和数据子帧
65
RB交换控制器
空分交换控制器

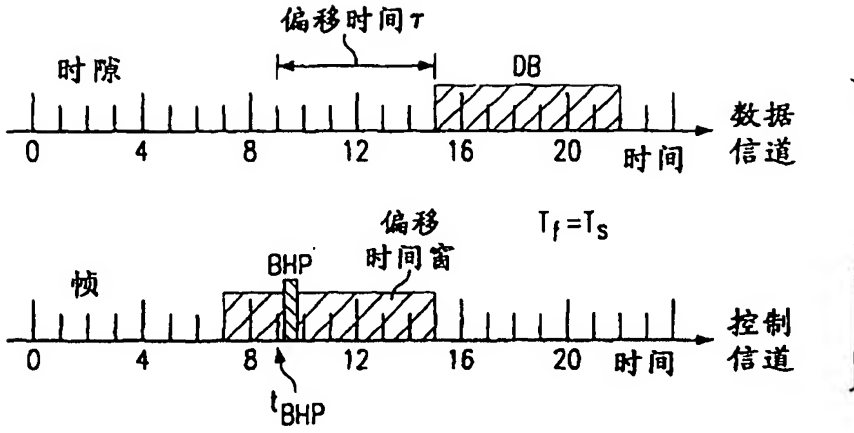

偏移时间τ
时隙
DB
0
4
8
12
16
20
时间
数据信道
偏移时间窗
$T_f=T_s$
BHP
帧
控制信道
t_{BHP}

图 4a

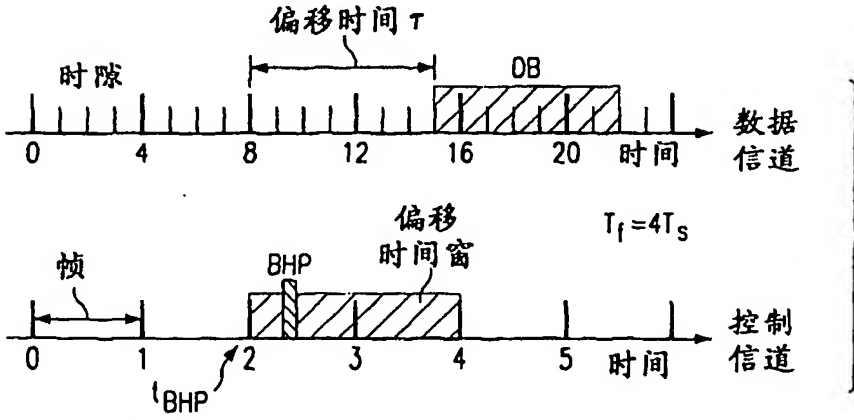
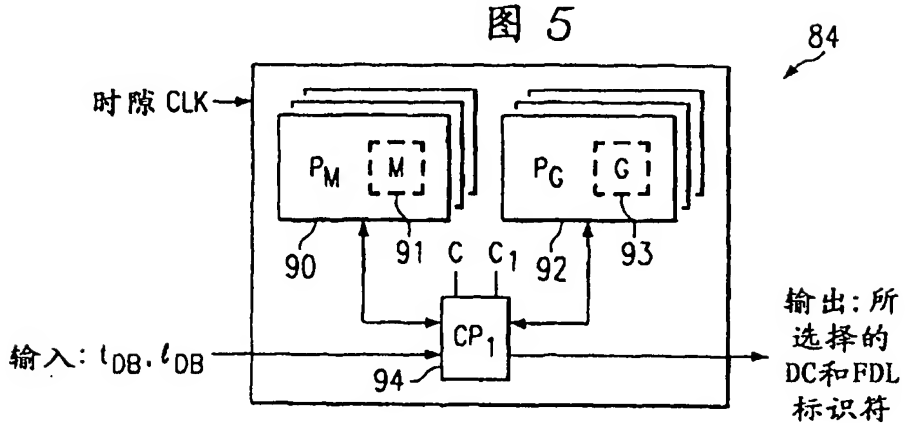

偏移时间τ
时隙
DB
0
4
8
12
16
20
时间
数据信道
偏移时间窗
$T_f=4T_s$
帧
BHP
0
1
2
3
4
5
时间
控制信道
t_{BHP}

图 4b

图 5


84
时隙 CLK
P_M
M
P_G
G
90
91
C
C_1
92
93
CP_1
94
输入：t_{DB}, t_{DB}
输出：所选择的DC和FDL标识符

91

	未调度时刻	信道ID
0	t_0	H_0
1	t_1	H_1
2	t_2	H_2
3	t_3	H_3
4	t_4	H_4
5	t_5	H_5
6	t_6	H_6
7	t_7	H_7

图 6

93

	起始时间	结束时间	信道ID
0	l_0	r_0	H_0
1	l_1	r_1	H_1
2	l_2	r_2	H_2
⋮	⋮	⋮	⋮
G-2	l_{G-2}	r_{G-2}	H_{G-2}
G-1	l_{G-1}	r_{G-1}	H_{G-1}

图 7

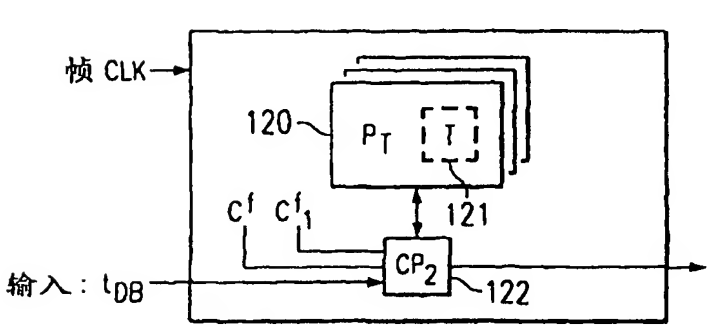


图 9

输出：在 t_{DB}, $t_{DB}+L_1$, $t_{DB}+L_2$ 时间在CC上可能的BHP出发时间

图 8

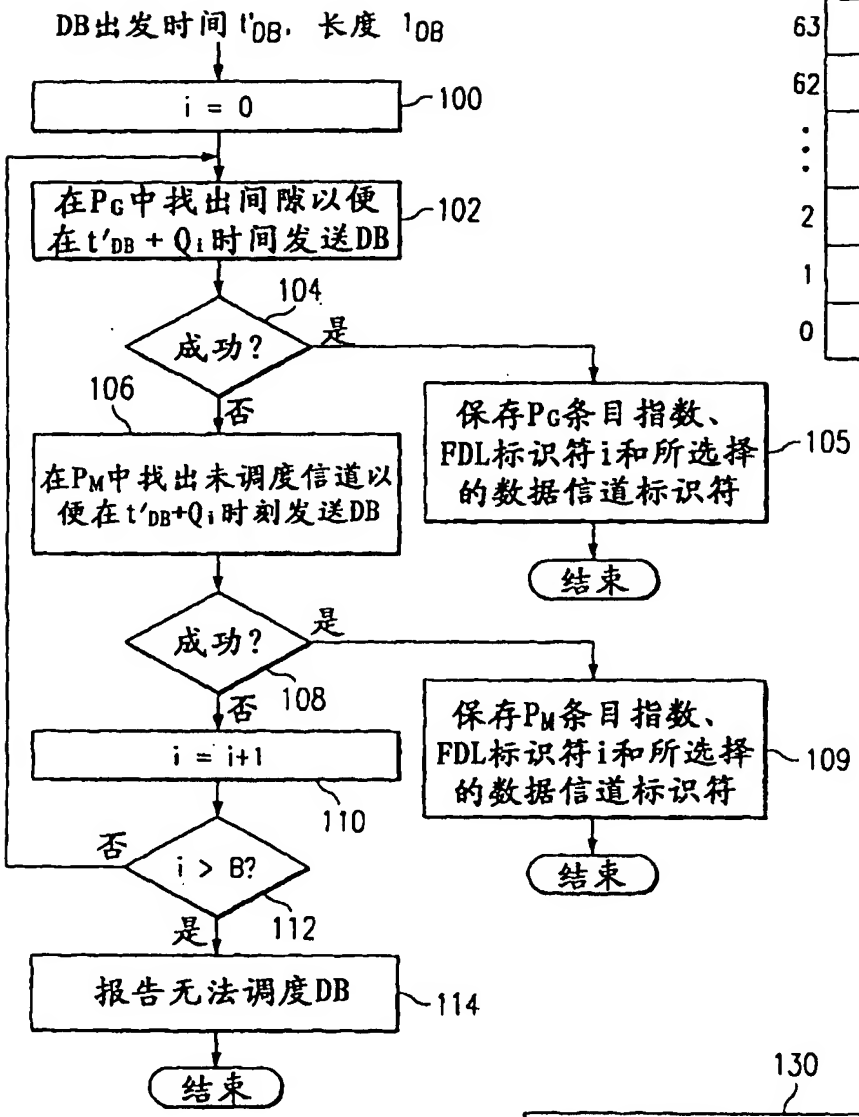
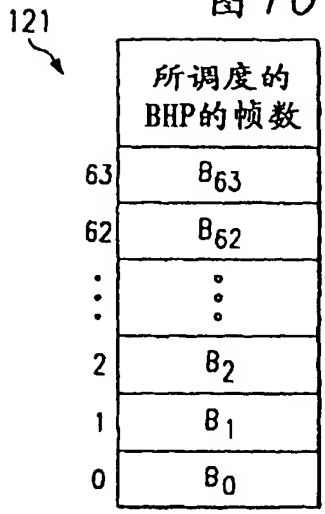

DB出发时间 t'_{DB}，长度 l_{DB}
i = 0
100
在P_G中找出间隙以便在$t'_{DB} + Q_i$时间发送DB
102
104
成功?
是
否
106
在P_M中找出未调度信道以便在$t'_{DB}+Q_i$时刻发送DB
保存P_G条目指数、FDL标识符i和所选择的数据信道标识符
105
结束
成功?
是
否
108
保存P_M条目指数、FDL标识符i和所选择的数据信道标识符
109
i = i+1
110
结束
否
i > B?
是
112
报告无法调度DB
114
结束

图 10


121
所调度的BHP的帧数
63
B_{63}
62
B_{62}
2
B_2
1
B_1
0
B_0

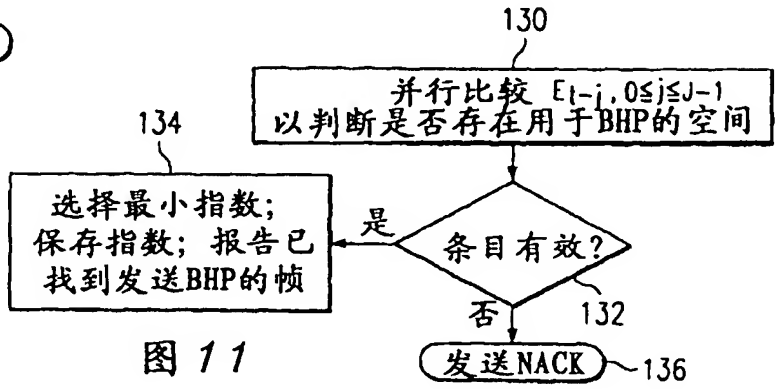
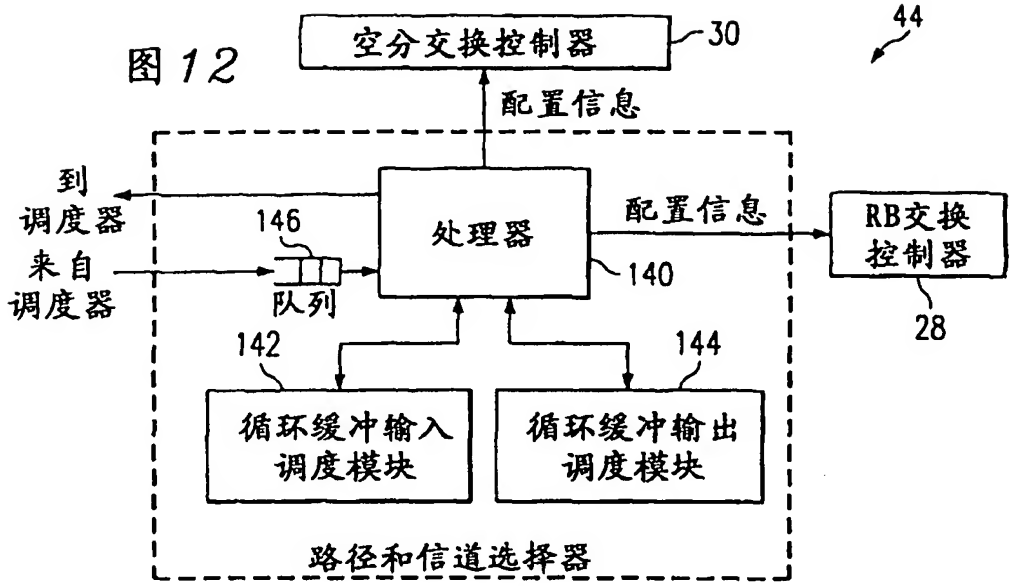

130
并行比较 $E_{i-j}, 0 \le j \le J-1$
以判断是否存在用于BHP的空间
134
选择最小指数;
保存指数; 报告已找到发送BHP的帧
是
条目有效?
否
132
发送NACK
136

图 11

图 12


空分交换控制器
30
44
配置信息
到
调度器
来自
调度器
146
队列
处理器
配置信息
RB交换
控制器
140
28
142
144
循环缓冲输入
调度模块
循环缓冲输出
调度模块
路径和信道选择器

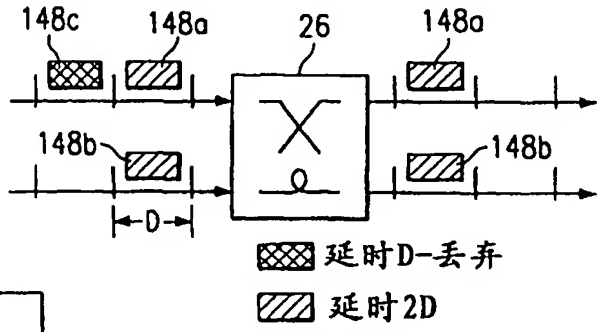

148c
148a
26
148a
148b
148b
D
延时D-丢弃
延时2D

图 13

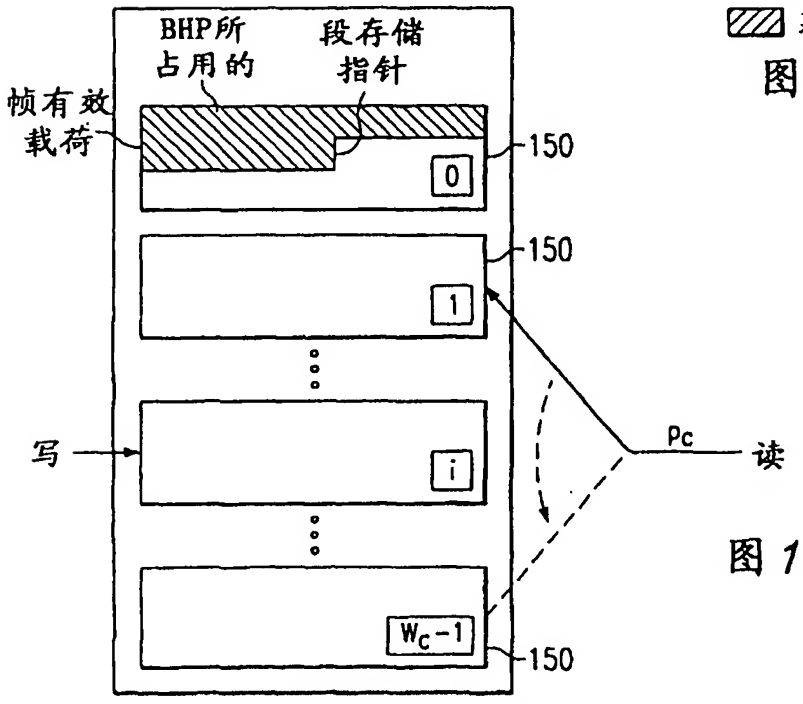
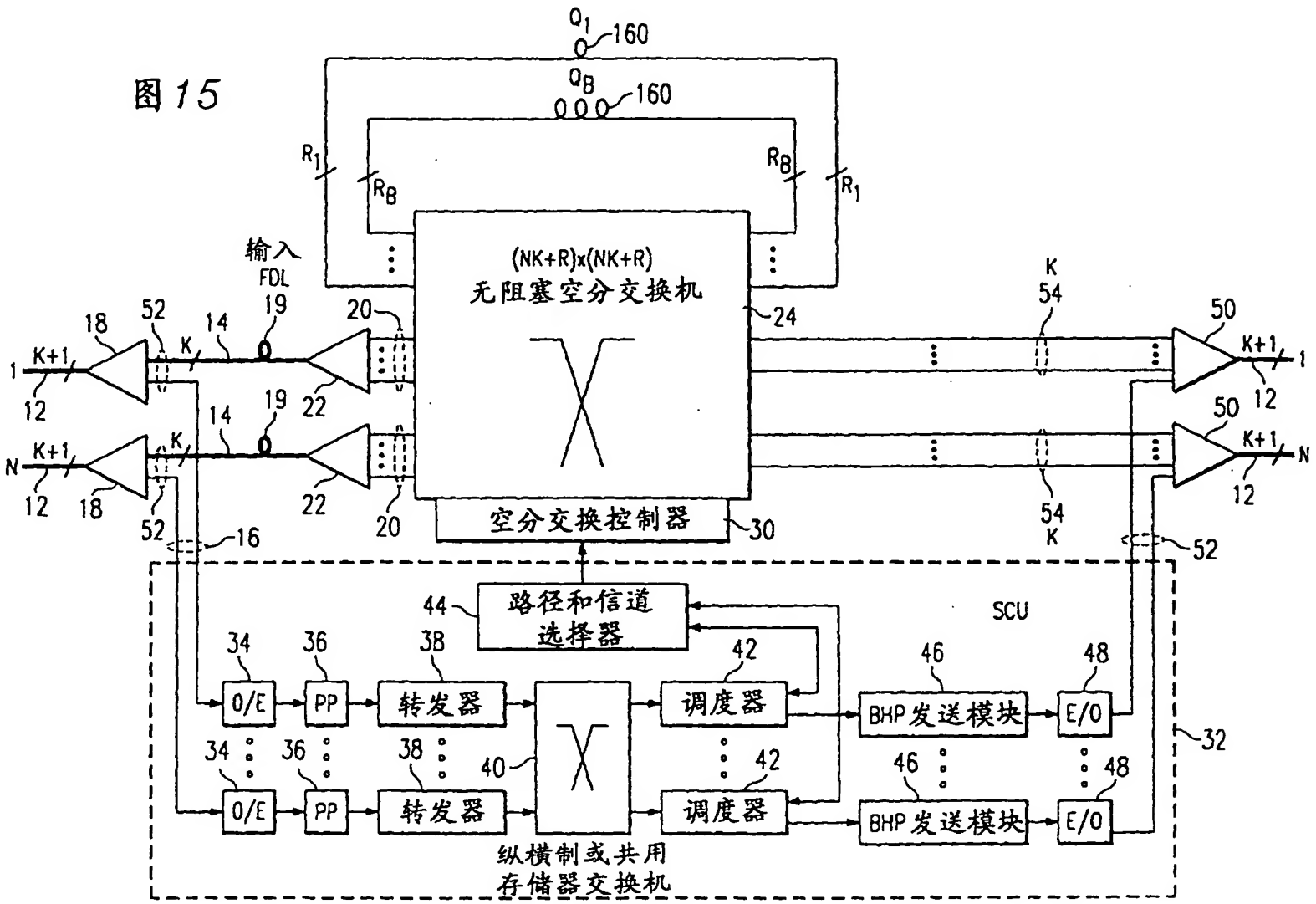

BHP所
占用的
段存储
指针
帧有效
载荷
0
150
1
150
写
i
Pc
读
Wc-1
150

图 14

图15


Q_1 160
Q_B 160
R_1
R_B
R_B
R_1
输入
FDL
(NK+R)x(NK+R)
无阻塞空分交换机
空分交换控制器
30
24
K
54
K
54
50
K+1
1
12
50
K+1
N
12
18
52
K
14
19
20
22
K+1
1
12
K+1
N
12
18
52
16
路径和信道
选择器
44
SCU
34
36
38
42
46
48
O/E
PP
转发器
调度器
BHP发送模块
E/O
40
32
纵横制或共用
存储器交换机

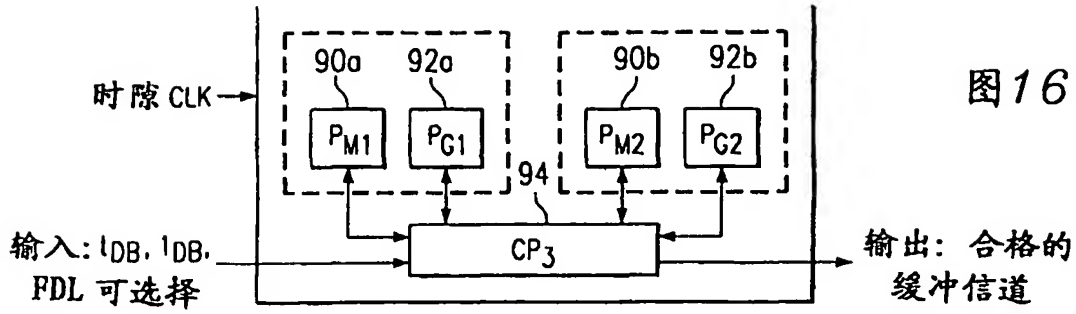


图16

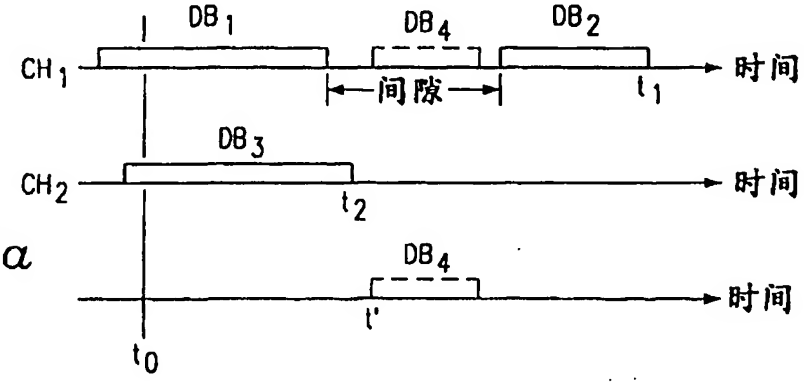


图17a

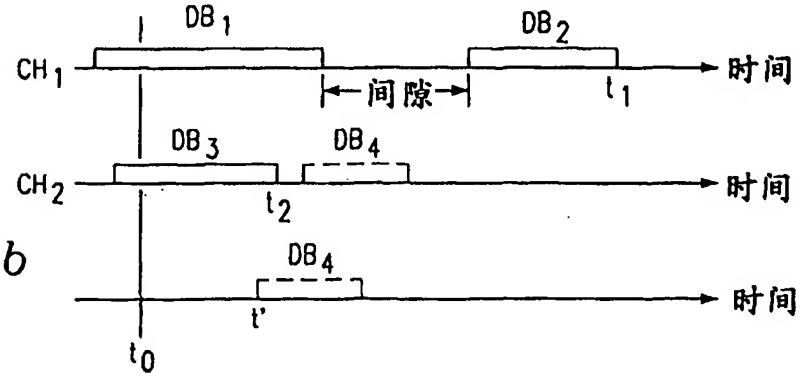


图17b

图18

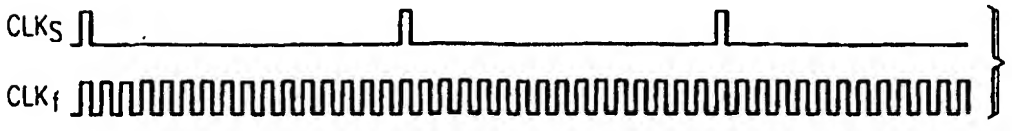


图19a

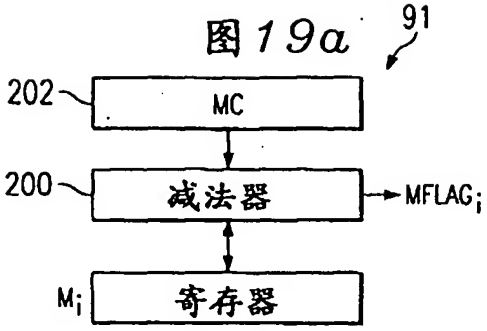


图19b

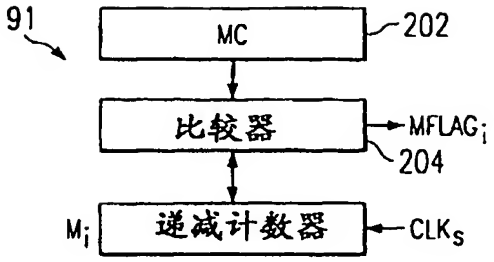


图20

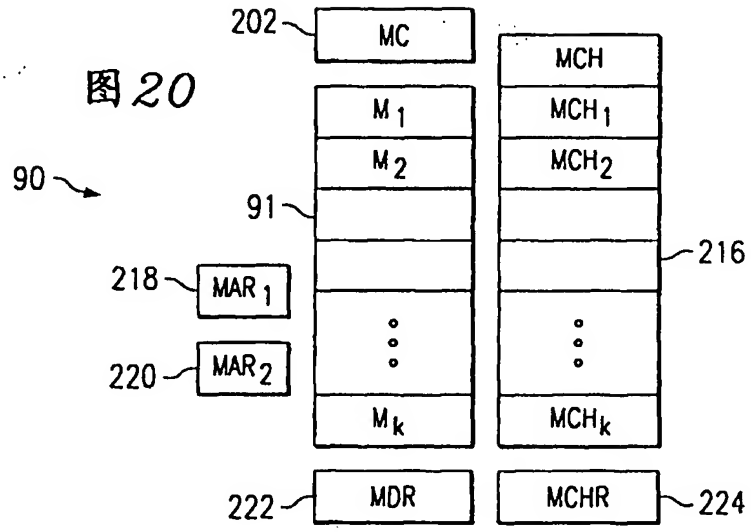
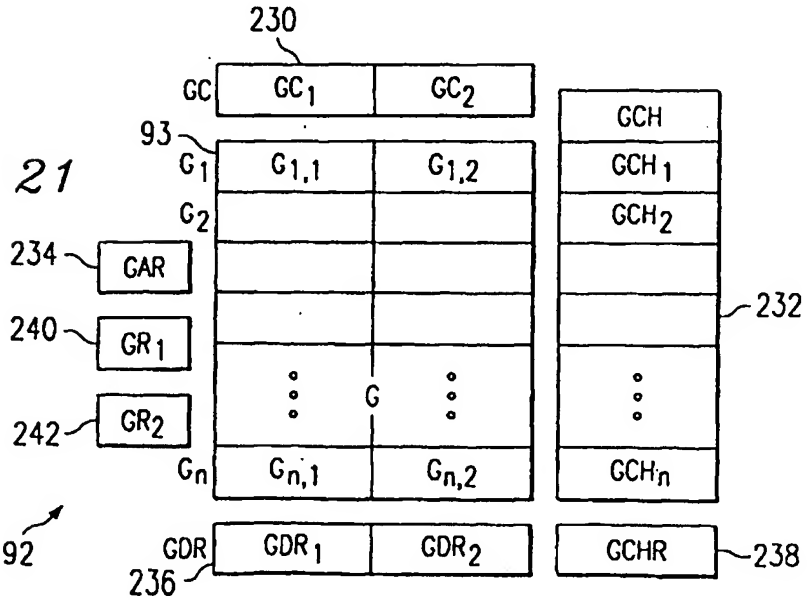


图21



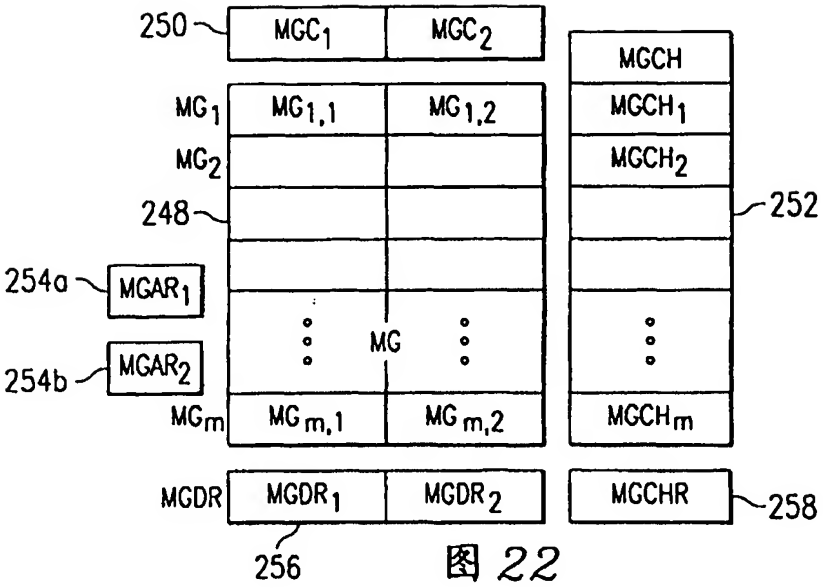

250
MGC1
MGC2
MGCH
MG1
MG1,1
MG1,2
MGCH1
MG2
MGCH2
248
252
254a
MGAR1
MG
254b
MGAR2
MGm
MG m,1
MG m,2
MGCHm
MGDR
MGDR1
MGDR2
MGCHR
258
256

图 22

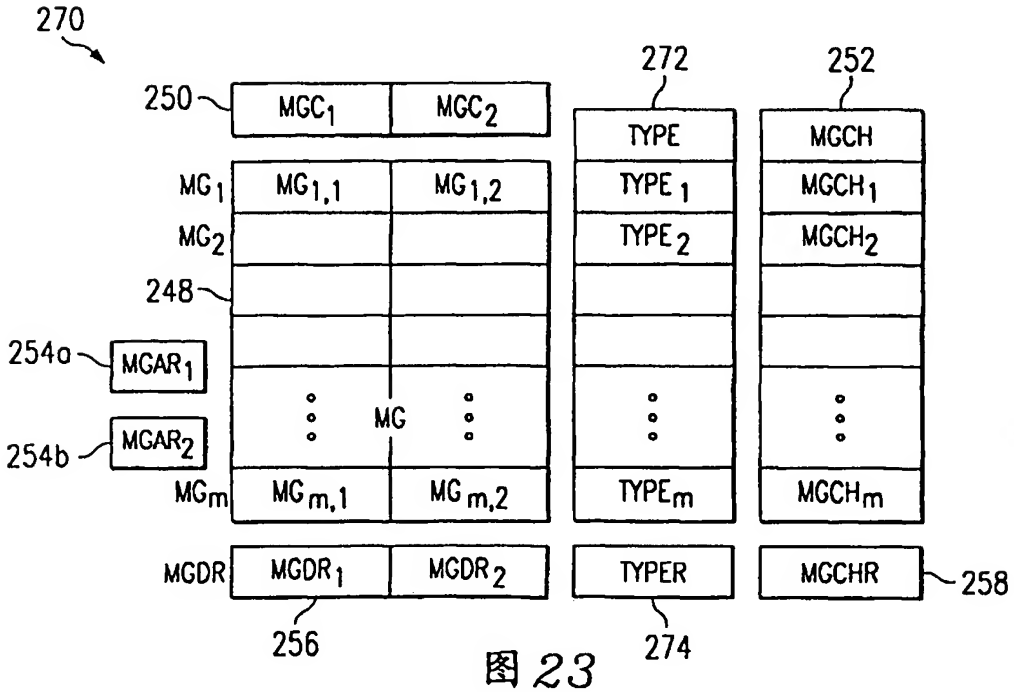
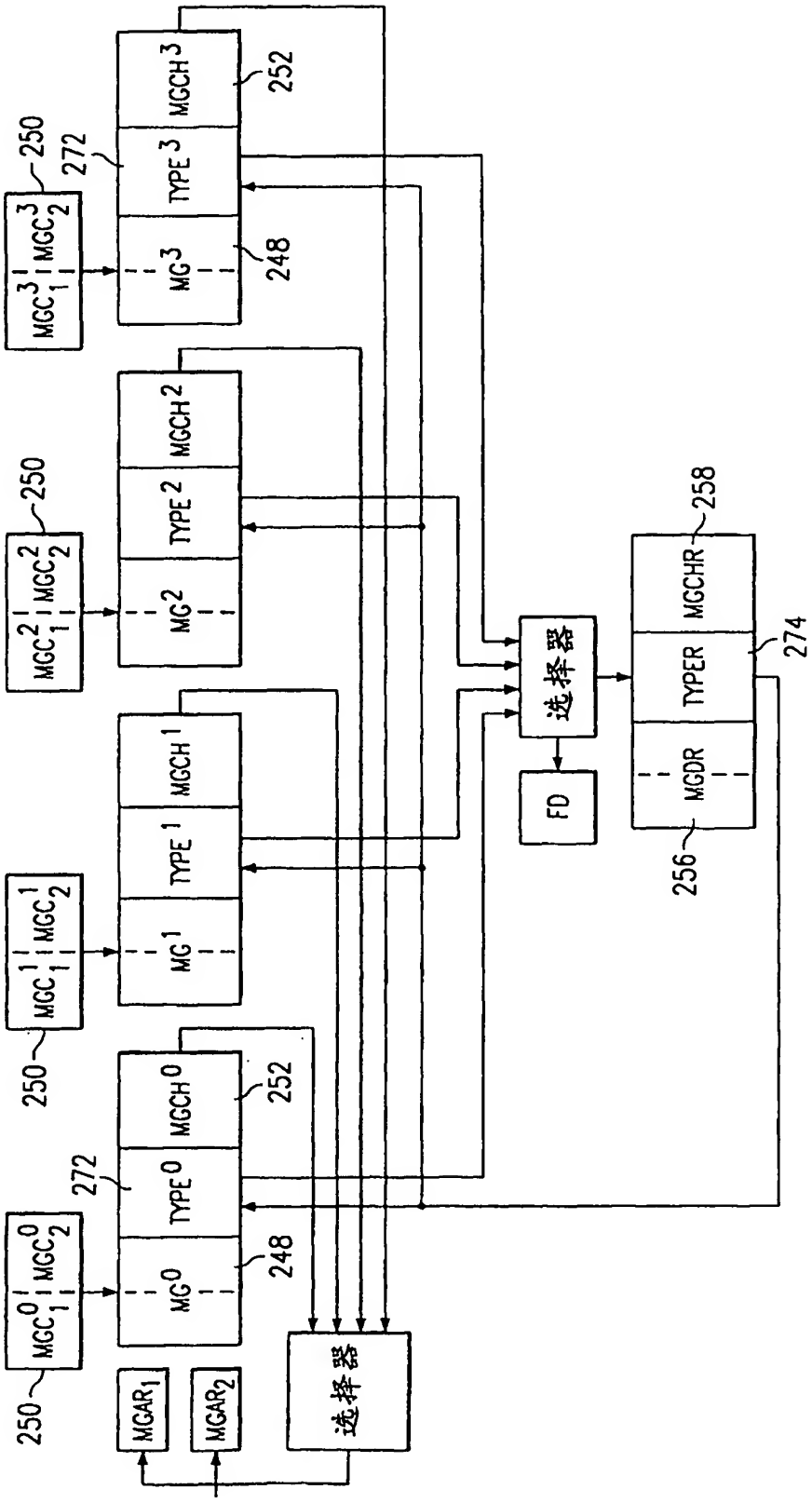

270
272
252
250
MGC1
MGC2
TYPE
MGCH
MG1
MG1,1
MG1,2
TYPE 1
MGCH1
MG2
TYPE2
MGCH2
248
254a
MGAR1
MG
254b
MGAR2
MGm
MG m,1
MG m,2
TYPEm
MGCHm
MGDR
MGDR1
MGDR2
TYPER
MGCHR
258
256
274

图 23

图 24


MGC$^0_1$ MGC$^0_2$
MG^0
TYPE^0
MGCH^0
MGC$^1_1$ MGC$^1_2$
MG^1
TYPE^1
MGCH^1
MGC$^2_1$ MGC$^2_2$
MG^2
TYPE^2
MGCH^2
MGC$^3_1$ MGC$^3_2$
MG^3
TYPE^3
MGCH^3
250
252
248
272
MGAR$_1$
MGAR$_2$
选择器
FD
选择器
MGDR
TYPER
MGCHR
256
258
274

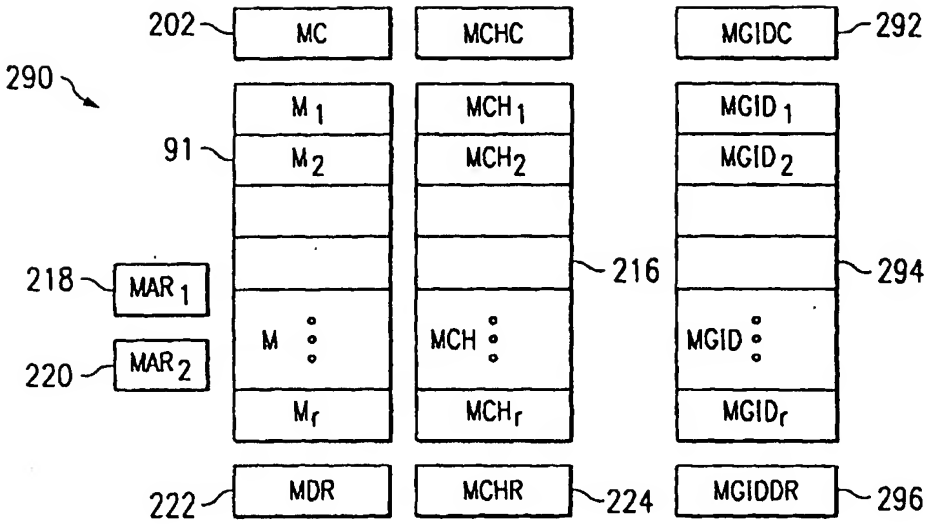

202
MC
MCHC
MGIDC
292
290
M 1
MCH 1
MGID 1
91
M 2
MCH 2
MGID 2
216
294
218
MAR 1
220
MAR 2
M
MCH
MGID
M r
MCH r
MGID r
222
MDR
MCHR
224
MGIDDR
296

图25

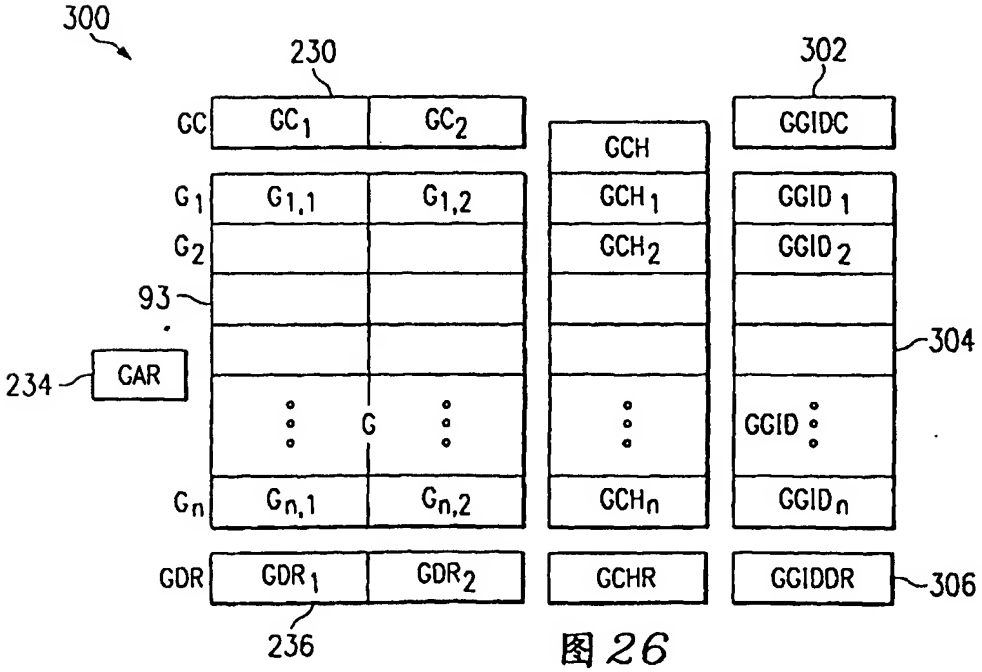

300
230
302
GC
GC 1
GC 2
GCH
GGIDC
G 1
G 1,1
G 1,2
GCH 1
GGID 1
G 2
GCH 2
GGID 2
93
304
234
CAR
G
GGID
G n
G n,1
G n,2
GCH n
GGID n
GDR
GDR 1
GDR 2
GCHR
GGIDDR
306
236

图26